AD-A033 480   PRC INFORMATION SCIENCES CO MCLEAN VA   F/G 17/9

PROGRAM MAINTENANCE MANUAL FOR THE REFERENCE SCENE SOFTWARE (RS--ETC(U)

OCT 76 C B SHELTON   DAAK02-75-C-0098

UNCLASSIFIED   PRC-R-1939   ETL-0067   NL

1 OF 2
AD
A033480

ADA033480

ETL-0087

PROGRAM MAINTENANCE MANUAL
FOR THE
REFERENCE SCENE SOFTWARE (RSS)

PRC INFORMATION SCIENCES COMPANY
7600 Old Springhouse Road
McLean, Virginia 22101

15 October 1976

COPY AVAILABLE TO DDC DOES NOT
PERMIT FULLY LEGIBLE PRODUCTION

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

D D C
RECEIVED
DEC 16 1976
F

Prepared for

U.S. ARMY ENGINEER TOPOGRAPHIC LABORATORIES
Fort Belvoir, Virginia 22060

1473

Copy available to DDC does not
permit fully legible reproduction

Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The citation in this report of trade names of commercially available products does not constitute official endorsement or approval of the use of such products.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ETL-0067	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): PROGRAM MAINTENANCE MANUAL FOR THE REFERENCE SCENE SOFTWARE (RSS)		5. TYPE OF REPORT & PERIOD COVERED: Contract Report
		6. PERFORMING ORG. REPORT NUMBER: PRC-R-1939
7. AUTHOR(s): Ms. Carolyn D. Shelton		8. CONTRACT OR GRANT NUMBER(s): DAAK02-75-C-0098
9. PERFORMING ORGANIZATION NAME AND ADDRESS: PRC Information Sciences Co. 7600 Old Springhouse Rd McLean, VA 22101		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 169p.
11. CONTROLLING OFFICE NAME AND ADDRESS: U.S. Army Engineer Topographic Laboratories Fort Belvoir, Virginia 22060		12. REPORT DATE: 15 Oct 76
		13. NUMBER OF PAGES: 167
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report): UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Copy available to DDC does not permit fully legible reproduction

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Reference Scene Software (RSS)
Radar
Radar Simulation
Computer Simulation
Automated Cartography
Terminal Guidance
Topographic Data Bases

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Reference Scene Software (RSS) is a set of eleven CDC 6400 computer programs used in-house at the U.S. Army Engineer Topographic Laboratories (USAETL), Ft. Belvoir, Virginia, to produce simulated Plan Position Indicator (PPI) radar scenes. The two inputs required by RSS are a matrix array (raster format) of digital terrain elevations and a corresponding vector digitized list of planimetry features (roads, lakes, railroads, cities, rivers, etc.). The output of RSS is a raster format magnetic tape image of the circular PPI scene, over

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 407 126

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

which is later formatted onto 35mm film and machine compared to the actual PPI scene of the area to determine the "goodness" of correlation.

These programs were originally developed by the Naval Training Equipment Center (NTEC), Orlando, Florida, for visual flight simulation. They were converted to run on the ETL CDC 6400 computer, new input and output routines were developed, and the radar modeling algorithm was changed to produce a better machine readable rather than better human readable scene.

RSS is being used to determine the data base input requirements and the radar modeling algorithm parameters necessary for producing "correlatable" reference scenes.

ABSTRACT

PRC Information Sciences Company Report R-1939
Program Maintenance for the Reference Scene Software
Carolyn B. Shelton, October 1976 (Unclassified)

The U.S. Army Engineer Topographic Laboratories (ETL) at Fort Belvoir is presently engaged in a concentrated effort aimed at developing a methodology for generating radar reference scenes from raw cartographic data. A central part of this effort is the identification of the minimum set of radar attributes required in such reference scenes in order that they provide sufficient correlation when compared with actual PPI radar images. The objective of the effort described herein is the development of Reference Scene Generation Software (RSS) to be used by ETL as a research tool in the development of the final reference scene generation criteria.

The RSS is based on Digital Radar Landmass Simulation Software (DRLMS) provided by the Naval Training Equipment Center at Orlando, Florida. The first step in the development of RSS was to convert these programs to run in-house on ETL's CDC 6400. New input routines were written to permit the use of in-house data bases, and a new output routine was written to permit the radar scene output to be displayed on ETL's DICOMED plotter.

The second phase of the program involved further modifications of the programs to make them more suitable for correlation work. Among the improvements added were the capabilities to vary image resolution, size and coloring. The software was also analysed and corrected to improve its geometric accuracy.

Finally, a routine was added to permit the incorporation of the altitude layover effect into the reference scenes. This effect produces a nonuniform radial translation of the points on the radar image and its inclusion is expected to improve the correlation obtainable with the reference scenes.

ACCESSION for
NTIS White Section ☑
DDC Buff Section ☐
UNANNOUNCED ☐
JUSTIFICATION ....
BY ....
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

Copy available to DDC does not permit fully legible reproduction

PREFACE

This work was principally performed by Ms. Carolyn B. Shelton, Planning Research Corporation, under contract number DAAK02-75-C-0098, Radar Programs Conversion for the U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia, Bruce B. Zimmerman, Contracting Officer's Technical Representative.

This document is a Program Maintenance Manual for the Reference Scene Software (RSS) presently being used by the USAETL in their radar scene correlation studies. As such, it provides sufficient information to enable the programmer to perform such software modifications as may be required during the continuing development of the system.

This document should be used in conjunction with the RSS User's Manual (ETL-066). NR

RSS is based on Digital Radar Landmass Simulation Software supplied by the Naval Training Equipment Center, Orlando, Florida.

TABLE OF CONTENTS

1. General Description

1.1 System Application. ETL is presently engaged in a concentrated effort aimed at developing a methodology for generating radar reference scenes from raw cartographic data. A central part of this effort is the identification of the minimum set of radar attributes required in such reference scenes in order that they provide sufficient correlation when compared with actual PPI radar images. RSS is being used by ETL as a research tool in the development of such a set of reference scene generation criteria.

1.2 Equipment Environment. The software described herein is written in FORTRAN IV for a CDC 6400 large-scale computer using the SCOPE 3.4 operating system. Because of the inherently large storage requirements of the programs, on-line disk storage is used extensively.

A DICOMED Model 36 scanner/plotter is presently being used as the output display device.

2. Detailed Software Description

2.1. System Description

The radar scene simulation process employed by this system consists of four subsystems: (1) the planimetry preparation subsystem, (2) the terrain preparation subsystem, (3) the scene generation subsystem, and (4) the image processing subsystem.

Programs RSS1 through RSS5 comprise the planimetry preparation subsystem.

The input to RSS1 is a raw planimetry data file which contains the X-Y pairs defining the boundary of each feature and a feature code identifying the feature type (e.g., road, lake, etc.). This information is recorded by the ETL Bendix Programmable Digitizer. RSS1 unpacks the feature information, throwing away the digitizer command codes, and outputs the unpacked feature data onto a disk file. This disk file is the input to program RSS2.

In RSS2 the feature data is converted from X-Y pairs defining a feature perimeter, to horizontal strip data. This procedure result is a file with enclosed feature data. Each horizontal strip is defined by the X, Y coordinates of its left end, and by a delta-X value which is the strip length.

Because of limited core memory available, it is necessary to divide the map into smaller regions for processing. Program RSS3 does this by dividing the planimetry strips generated by RSS2 at fixed region boundaries.

Program RSS4 is a sorting routine which reorders the planimetry data generated by RSS3. The output of RSS4 is an equivalent data base in

which the data is ordered by the following priorities respectively:

1 - region number
2 - merge priority
3 - Y coordinate
4 - X coordinate

Program RSS5 merges the planimetry data, output from RSS4, in accordance with the above priority list. The output of RSS5 is a file in which the contents of each region are described by non-overlapping horizontal strips. In order to accomplish this, strips from low merge priorities (such as cities) are written first and are written over where necessary by strips from features with higher merge priorities (such as lakes). The completion of this program marks the end of the planimetry preparation subsystem.

Program RSS6 is the terrain preparation program. The terrain data is divided into regions with the same boundaries as that generated by RSS3.

Programs RSS7 through RSS9 comprise the scene generation subsystem.

Program RSS7 merges the planimetry and terrain files for those regions lying within the radar ground range of the target. The output from RSS7 consists of one record for each region within the radar range of the target. Each record consists of an appropriate header, followed by the planimetry/terrain content of each point in the region.

Program RSS8 uses the cartesian data base output from RSS7 to construct a series of radial scan lines required to simulate a Plan-Position-Indicator (PPI) radar scene. This operation consists of cartesian-to-polar coordinate transformation.

The output from RSS8 consists of N scan lines where $360^{\circ}/N$ is the angular increment between scan lines. The single record for each scan line contains the planimetry/terrain data for all points lying along the scan line, beginning at the target location and ending 20 nautical miles out.

Program RSS9 applies the radar effects to the radial scan line data from RSS8. The effects considered are discussed later under the program description for RSS9. This program also calls subroutines which scale the image size and convert the data back to cartesian coordinates. Program RSS9 outputs a data record for each cartesian coordinate to a disk file. RSS9 marks the end of the scene generation subsystem.

At this point the radar scene is complete. The image processing subsystem begins with the SORT program which orders the data records generated by RSS9 by Y coordinate and X coordinate.

Program RSS10 formats the final image data onto an output tape using the cartesian coordinate data from program SORT. This output tape is then read into a DICOMED plotter which outputs the radar image.

Operationally this simulation system generates four scenes for each target. Each scene depicts the radar image of a target as viewed from a different altitude. Programs RSS1 through RSS8 need only be run once for any given target. Programs RSS9 through RSS10, however, need to be run once for every scene generated.

The flow diagram for this system is illustrated in Figure II.1. (I.1 in User's Manual)

2.2 Program Description

This section gives a detailed description of each routine and subroutine in this system. Attachments for each program described include

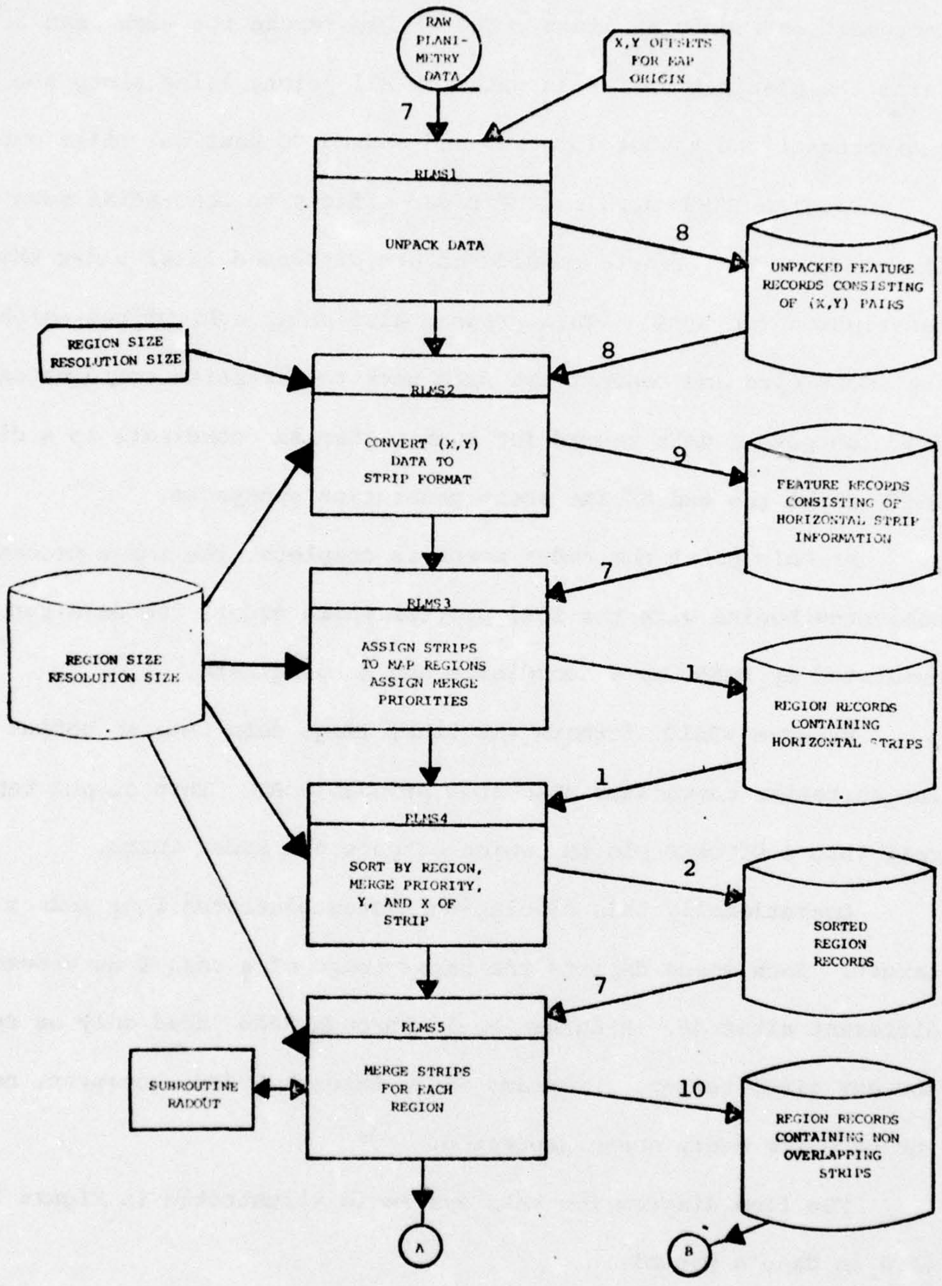


FIGURE II.1 - DRLMS PROGRAM AND DATA FLOW
(PAGE 1 OF 2)

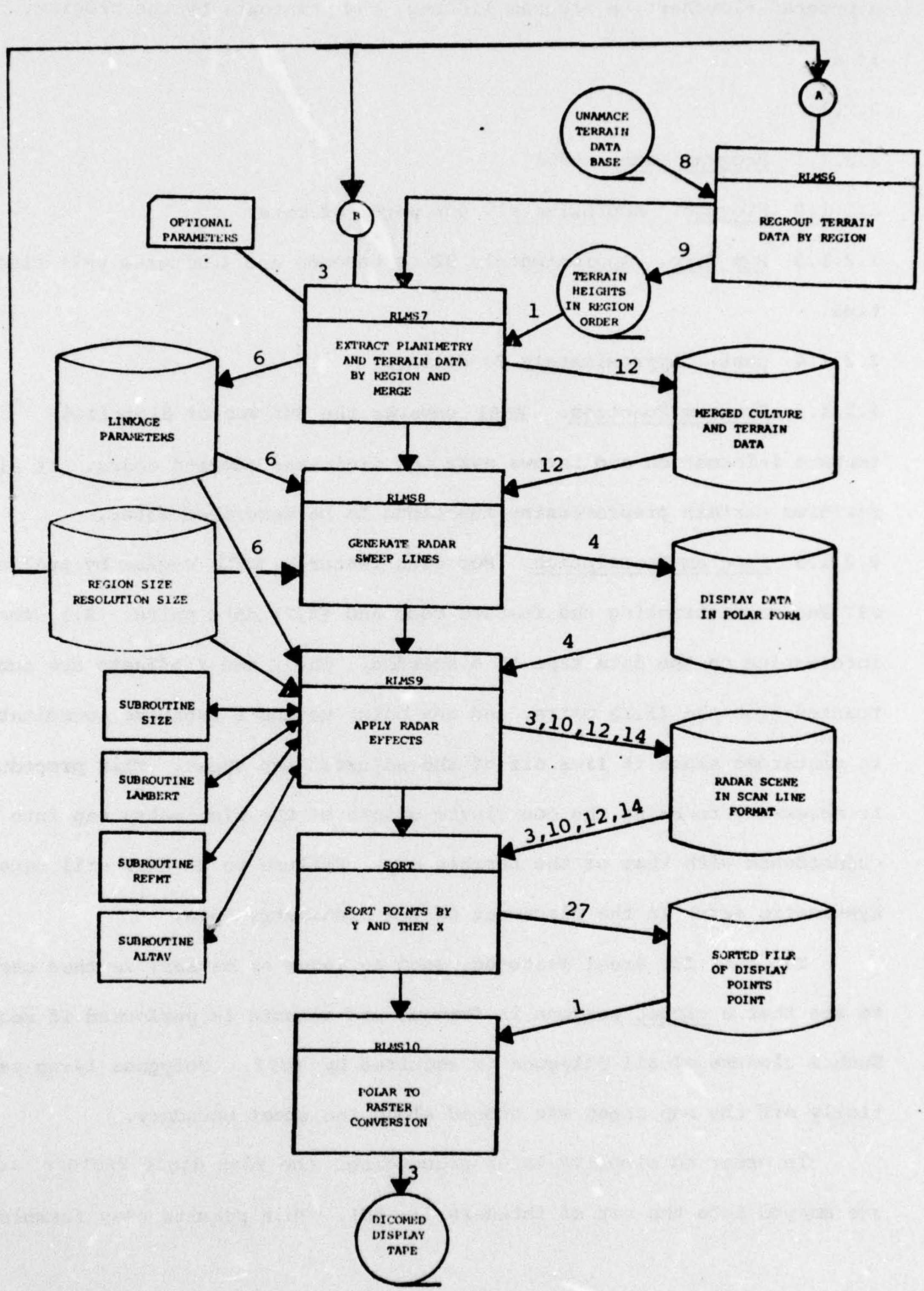


FIGURE II.1 - DRLMS PROGRAM AND DATA FLOW
(PAGE 2 OF 2)

a program flowchart, a program listing, and printouts by the program, if any.

2.2.1

2.2.1.1 Program Name. RSS1

2.2.1.2 Storage. Approximately 43K words of core.

2.2.1.3 Run Time. Approximately 92 cp seconds and 4 minutes wall clock time.

2.2.1.4 Cost. Approximately 24 dollars.

2.2.1.5 Program Function. RSS1 unpacks the ETL vector digitized feature information and throws away the digitizer command codes. It also performs certain preprocessing functions to be described later.

2.2.1.6 Program Description. For each feature, RSS1 begins by pulling off and reconstructing the feature code and (X,Y) data pairs. All other information on the data tape is discarded. The X and Y offsets are subtracted from the (X,Y) pairs, and any point having a negative coordinate is discarded since it lies off of the adjusted map sheet. This procedure is necessary to bring the coordinate origin of the planimetry map into coincidence with that of the terrain map. Failure to do this will cause a systematic error in the placement of the planimetry data.

The data for areal features, such as lakes or cities, is then checked to see that a closed polygon is formed, and closure is performed if required. Such a closure of all polygons is required by RSS2. Polygons lying partially off the map sheet are closed along the sheet boundary.

In order to simplify later processing, the five digit feature codes are mapped into the set of integers (1, 60). This permits easy formulation

of translation tables which match feature codes with feature attributes such as merge priority and intrinsic radar return intensity.

Finally, RSS1 does special processing on certain features. If a feature is a power line the towers are separated from the cable and each piece is written out as a separate feature. A wide river is represented by two separate features, one defining its right bank and another defining its left bank. In addition to maintaining the two banks, RSS1 combines them into a third feature which is a closed polygon representing the water surface.

The following is a description of the important variables in RLMS1.

Label	Type	Description
Blank	Integer	Contains BCD representation for a blank character.
Block	Integer	Block counter.
Buffer	Integer Array	Digitizer input tape 200-word feature record
Down	Integer	Contains BCD representation for the character "D" which represents the down pen command (start of a line).
Etlif	Integer Array	Array containing the ETL feature codes
Finish	Integer	Contains BCD representation for the character "B" which indicates the end of block .
Header	Integer	Header information indicator. Contains either the BCD representation for "Yes" or "No".
IByte	Integer	Byte pointer into input buffer.
IClose	Integer	Closed feature indicator. Contains either the BCD representation for "Yes" or "No".

IEXP	Integer	Exponent of the base 10 required to convert BCD character to numeric machine code.
IF	Integer	Feature code
IMASK	Integer	Six bit mask
IN	Integer	Word counter for output buffer record
INDIC	Integer	Output unit device check word
INDI	Integer	Input unit device check word
INP1	Integer	Value of 1N+1
INP2	Integer	Value of 1N+2
INP3	Integer	Value of 1N+3
INTAP	Integer	Input unit device number. INTAP=7
IN3	Integer	Number of coordinates (points *2) on right river bank
IN3P3	Integer	Value of IN3+3
IN4	Integer	Number of coordinates (points *2) on left river bank
INP4P3	Integer	Value of IN4+3
ISEQ	Integer	Feature counter
ISHIFT	Integer	Number of bits to shift word
IT	Integer	Value of IN4+IN3+2 which equal the total number of coordinates (points *2) in the left and right river banks
ITEMP	Integer	Temporary storage word
ITYPE	Integer	BCD representation for current feature type
IWD	Integer	Number of words in input record. Initialized at 180 words

KFLAG	Integer	River bank processing flag. 1= one bank read 2=both banks read
KMAX	Integer	Total number of words in output record
KOUNT	Integer	Byte counter into next block
LOC	Integer	Word pointer for input buffer
NBYTE	Integer	Byte pointer for input buffer
NEWBLK	Integer	New block indicator. Contains BCD representation for either "Yes" or "No"
NO	Integer	Contains BCD representation for "No"
NOCLOSE	Integer Array	Contains array of feature codes for unclosed features
NWD	Integer	Word counter for output buffer
NWORD	Integer	Word counter for input buffer
OUTAP	Ingeger	Output unit device number. OUTAP=8
PEN	Integer	Pen command
POINT	Integer	BCD representation for a decimal point
RESTART	Integer	Restart indicator for picking up the remaining X or Y coordinate (BCD value) in the next block. Contains BCD representation for either "Yes" or "No"
START	Integer	Contains BCD representation for "(" which indicates start of new block
TEMP	Real	Temporary area for extracted X and Y coordinate character values
TYPE	Integer Array	Array containing BCD representation for English description of the feature type
UP	Integer	Contains BCD representation for the character "U" which represents the UP pen command (end of a line)

XMAX	Real	Maximum X coordinate value for this feature
XMIN	Real	Minimum X coordinate value for this feature
XORGIN	Real	X coordinate offset (correction factor)
X2	Real Array	Feature output buffer
X3	Real Array	Temporary storage for right river bank data
X4	Real Array	Temporary storage for left river bank data
YES	Integer	Contains BCD representation for "No"
YMAX	Real	Maximum Y coordinate value for this feature
YMIN	Real	Minimum Y coordinate value for this feature
YORGIN	Real	Y coordinate offset (correction factor)

2.2.1.7 Input. RSS1 requires one tape input file (tape 7) and one card input file. The tape input consists of a digitizer tape containing the (X,Y) coordinates of the points defining the location of the planimetry features on the map. The format of this tape is presented in Figure II.1

Each small box represents a 6-bit BCD character, and each physical record contains 1800 such characters (or 180 CDC words). The beginning of the information for each feature is marked by a "(". Following this are twelve characters which specify the starting location for the feature but are not used. The next five characters contain the feature code. A tabulation of the possible feature codes is given in Table II.1.

FEATURE TYPE	FEATURE CODE
RADAR GREY SHADES	20101 THRU 20320
LARGE RIVERS (WATER ON LEFT)	10110
LARGE RIVERS (WATER ON RIGHT)	10120
DAMS	10130
MARSHES AND SWAMPS	10140
LAKES	10150
ISLANDS	10160
RIVERS AND STREAMS	10170
RAILROAD YARDS	10210
RAILROADS	10220
TOWNS AND SUBURBS	10310
MEDIUM CITIES AND COMMERCIAL AREAS	10320
BIG CITIES AND INDUSTRIAL AREAS	10330
LARGE ISOLATED BUILDINGS	10340
INTERSTATE HIGHWAYS AND TURNPIKES	10410
MAJOR ROADS	10420
SECONDARY ROADS	10430
UNPAVED ROADS AND TRAILS	10440
AIRPORT	10450
POWER LINE TOWERS (WITH CABLES)	10510
DRIVE-IN MOVIES	10520
FIRE OR RADIO TOWERS	10530
CEMETERIES	10540
POL AREA	10550
HARDWOOD FOREST	10610
EVERGREEN FOREST	10620
MEADOWS AND GRASSY FIELDS	10630
DRY ROCKY AREAS	10640
SAND AND SAGEBRUSH AREAS	10650
SNOW COVERED AREAS	10660
DRY RIVERBEDS, CANALS, AND STORM DRAINS	10670
DRY LAKE BEDS AND GULCHES	10680

TABLE II.1 - CORRESPONDENCE BETWEEN FEATURE TYPES AND FEATURE CODES

Following the feature code is a pen-down command and this signals the beginning of the (X,Y) data.

The (X,Y) data for the feature begins in the twentieth character of the feature record. Each coordinate is in an F7.3 format. This means that each digit (as well as the decimal point) is represented as a separate character, with the maximum coordinate value being 999.999. All coordinates are given in inches relative to the table origin. The last Y-value for the feature is followed by a "U" which is the pen-up command from the digitizer.

A feature record may begin anywhere in a block of data, and may require more than one block. Similarly, a single block may contain the information for several features of small size. A feature record may be broken at any point except in the middle of the string of characters representing the feature code or a coordinate. A "B" is used to signal the end-of-information for a particular block.

The card input to RSS1 consists of a single card containing the X and Y offsets. These numbers are subtracted from every (X,Y) pair for every feature and are required to bring the origin of the planimetry data into coincidence with the origin used for the terrain data. The information is entered in a 2F10.3 format.

2.2.1.8 <u>Output</u>. RSS1 outputs the processed planimetry file to disk. Each feature is represented by a record with the following format:

<u>Word 1</u>: Feature code

<u>Word 2</u>: Total number of coordinates in the record i.e. twice the number of points N.

Word 3-word 2N+2: X,Y data

RLMS1 also outputs a printout identifying the type and location of all features on the map sheet. A sample of this printout is given in Figure II.1.

2.2.1.9 Externals. System routines called by RSS1 are LENGTH and UNIT which are mass storage I/O function routines. There are no subroutine calls from this program.

2.2.1.10 Error Conditions. There are two error conditions for RSS1. One error condition occurs when there is a format error found on the digitizer tape. The messages output for this error condition are as follows:

```
INCOMPLETE FEATURE HEADER IN BLOCK XXXX
RUNSTREAM ANALYSIS TERMINATED
                  OR
ILLEGAL PEN COMMAND IN BLOCK XXX
RUNSTREAM ANALYSIS TERMINATED
```

The octal data for the block containing the error is printed after each of the above two error messages.

The other error condition occurs when an error is found while reading or writing to or from the input tape or the output disk storage. The messages generated for this error condition are as follows:

```
I/O ERROR DETECTED AT BLOCK XXXX TOTAL WORDS
TRANSFERRED = XXX    BUFFER STATUS = XX
                  OR
OUTPUT ERROR DETECTED AT BLOCK XXXX TOTAL
WORDS OUTPUT = XXX    BUFFER STATUS = XX
```

2.2.1.11 Program Flowchart. The flow diagram for RSS1 is illustrated in Figure II.2.

2.2.1.12 Program Listing. The program listing for RSS1 is attached.

SEQUENCE NO.	FEATURE CODE	DESCRIPTION	CLOSURE	NO. OF POINTS	MINX	MAXX	MINY	MAXY
1	32	DRY LAKE	YES	466	32.601	33.201	15.224	17.215
2	32	DRY LAKE	YES	2023	27.600	33.182	5.750	12.771
3	32	DRY LAKE	YES	1650	23.991	25.487	.003	5.772
4	32	DRY LAKE	YES	1694	.296	4.213	4.231	8.050
5	32	DRY LAKE	YES	1736	.306	5.287	.528	4.975
6	31	CANAL	NO	564	.334	.779	16.691	23.976
7	22	LEFT BANK	NO	680	.235	.639	16.552	24.745
8	21	RIGHT BANK	NO	710	.295	1.051	16.467	24.913
9	23	RIVER FILL	YES	1391	.295	1.051	16.457	24.915
10	16	SML. RIVER	NO	174	3.254	3.940	25.616	29.306
11	16	SML. RIVER	NO	475	.264	3.319	27.865	22.883
12	16	SML. RIVER	NO	287	.272	2.681	27.051	27.390
13	16	SML. RIVER	NO	613	.335	4.297	25.712	27.524
14	16	SML. RIVER	NO	370	3.953	6.519	26.070	26.732
15	16	SML. RIVER	NO	245	5.277	6.151	27.200	28.359
16	16	SML. RIVER	NO	121	5.076	5.344	26.227	27.223
17	16	SML. RIVER	NO	493	.758	3.792	21.341	23.118
18	16	SML. RIVER	NO	497	1.786	5.425	20.542	21.535
19	16	SML. RIVER	NO	86	3.835	4.132	19.921	20.554
20	16	SML. RIVER	NO	238	7.211	8.009	26.765	28.520
21	16	SML. RIVER	NO	803	7.892	11.344	20.105	26.784
22	16	SML. RIVER	NO	370	8.431	9.973	26.174	28.809
23	16	SML. RIVER	NO	304	9.742	10.715	22.399	25.773
24	16	SML. RIVER	NO	479	13.637	14.872	25.376	29.951
25	16	SML. RIVER	NO	447	14.745	16.707	26.792	30.474
26	16	SML. RIVER	NO	555	12.740	13.439	20.669	26.068
27	16	SML. RIVER	NO	1004	16.121	25.382	23.920	30.237
28	16	SML. RIVER	NO	386	19.627	23.707	26.715	27.557
29	16	SML. RIVER	NO	358	19.949	23.819	26.662	27.125
30	16	SML. RIVER	NO	330	22.288	25.305	26.145	27.071
31	16	SML. RIVER	NO	106	23.837	24.546	25.709	26.273
32	16	SML. RIVER	NO	301	19.467	22.625	25.389	25.583
33	16	SML. RIVER	NO	456	17.097	21.777	24.315	25.032
34	16	SML. RIVER	NO	1078	13.362	23.659	21.331	24.438
35	16	SML. RIVER	NO	282	28.488	29.836	23.236	24.552
36	16	SML. RIVER	NO	424	29.445	31.826	19.821	23.236
37	16	SML. RIVER	NO	780	24.101	30.602	20.102	22.470
38	16	SML. RIVER	NO	584	24.733	30.593	19.323	20.568
39	16	SML. RIVER	NO	1278	22.458	28.464	5.398	18.948
40	16	SML. RIVER	NO	833	24.357	29.252	11.687	19.532
41	16	SML. RIVER	NO	321	29.256	31.919	13.395	16.637
42	16	SML. RIVER	NO	421	22.390	23.926	10.970	14.931
43	16	SML. RIVER	NO	412	22.596	25.705	9.491	12.183
44	16	SML. RIVER	NO	755	23.698	27.714	6.107	14.264
45	16	SML. RIVER	NO	416	19.048	20.752	10.643	13.000
46	16	SML. RIVER	NO	937	20.719	24.543	.006	10.853
47	16	SML. RIVER	NO	378	12.048	16.256	7.534	8.582
48	16	SML. RIVER	NO	314	12.630	15.655	5.445	7.049

FIGURE II.1 - SAMPLE PRINTOUT FROM RSS1

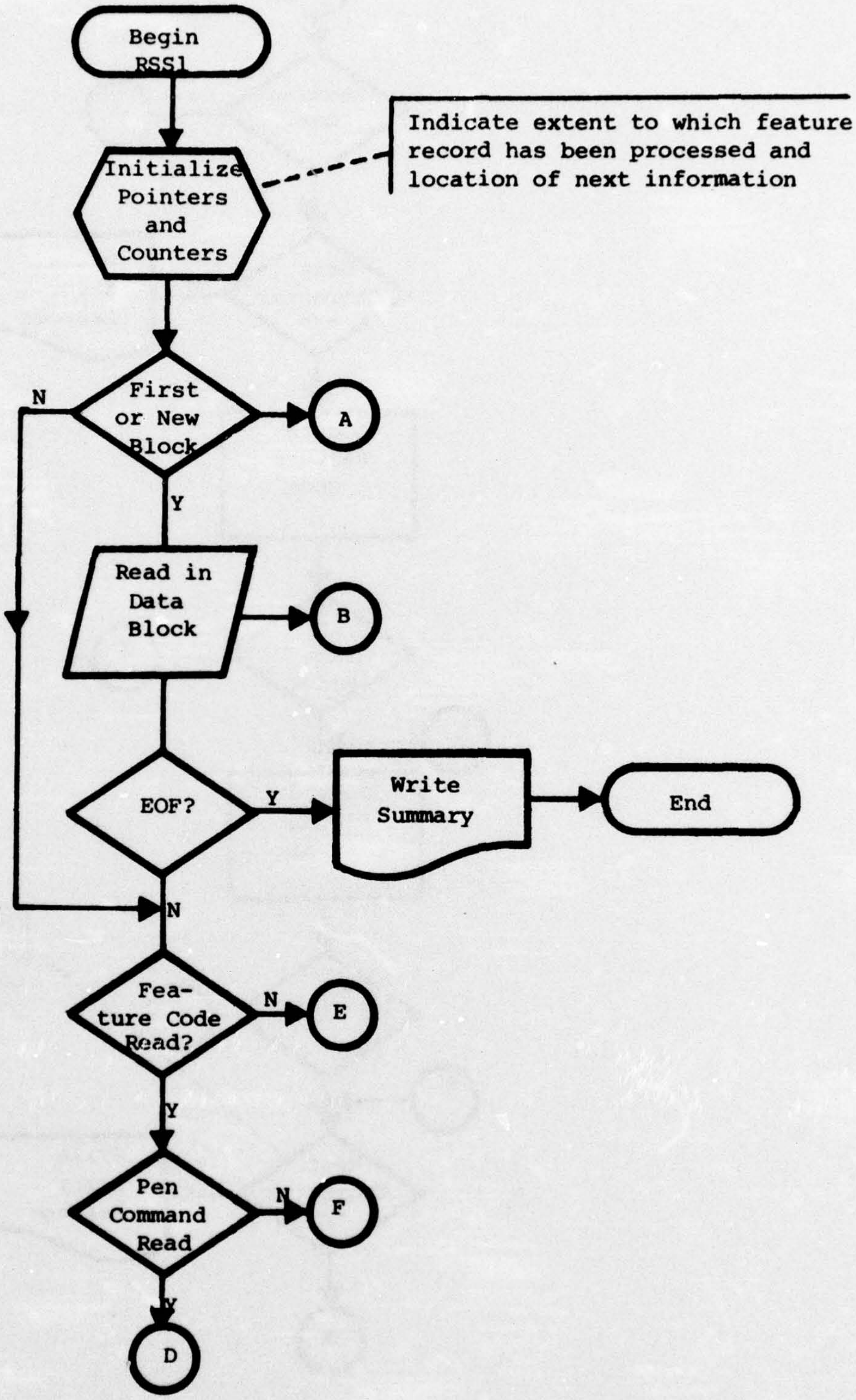


FIGURE II.2 - PROGRAM RSS1 FLOWCHART
(Page 1 of 4)

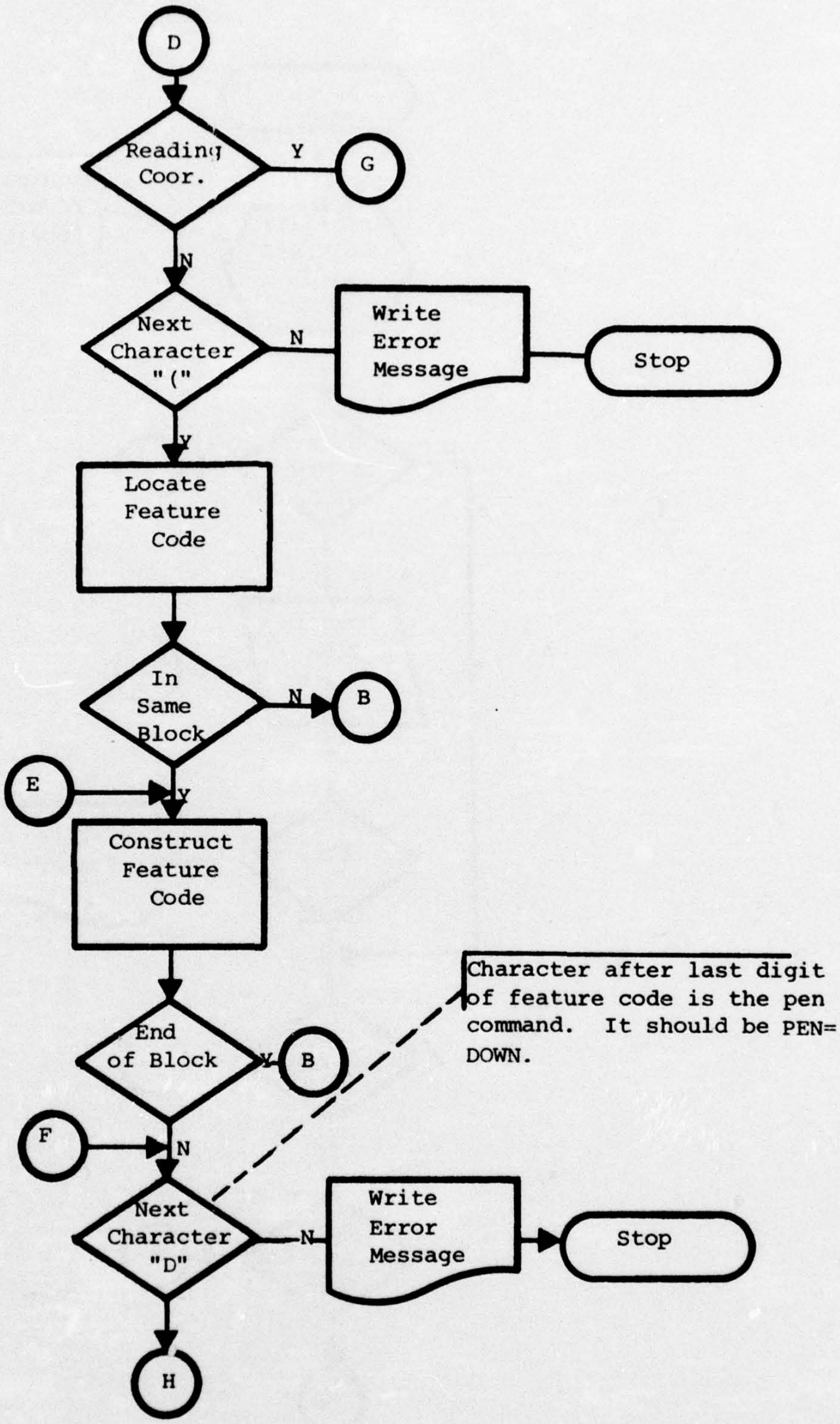


FIGURE II.2 - PROGRAM RSS1 FLOWCHART
(Page 2 of 4)

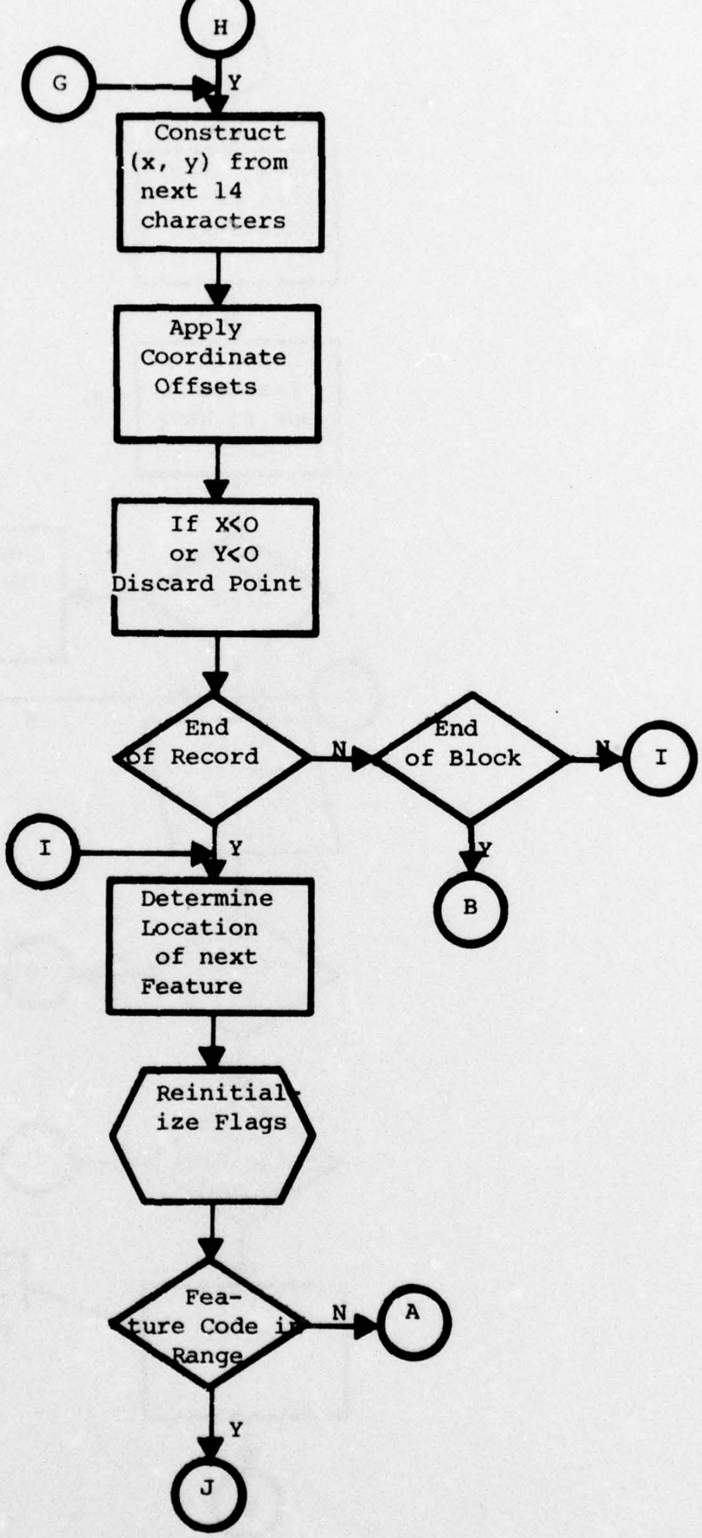


FIGURE II.2 - PROGRAM RSS1 FLOWCHART
(Page 3 of 4)

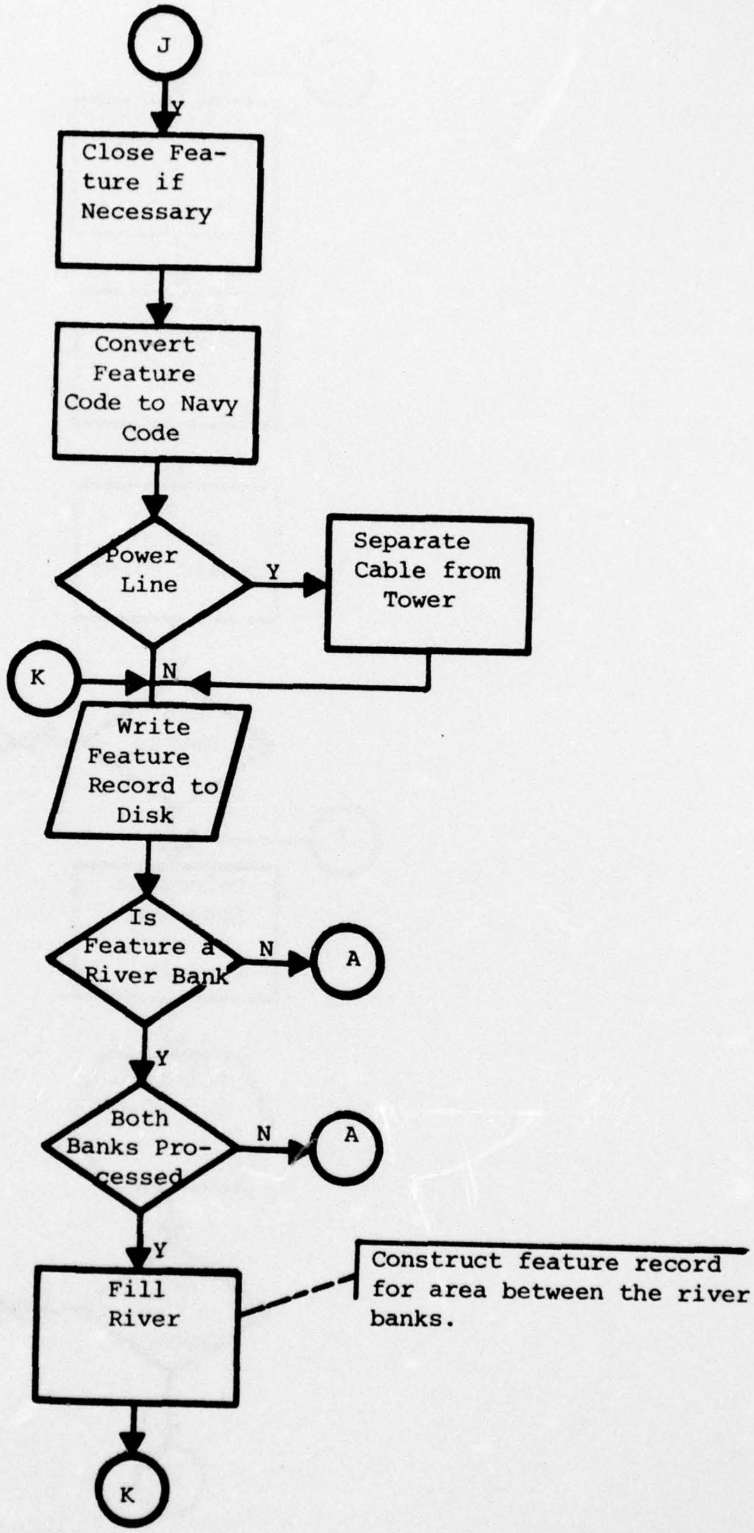


FIGURE II.2 - PROGRAM RSS1 FLOWCHART
(Page 4 of 4)

2.2.2

2.2.2.1 Program Name. RSS2

2.2.2.2 Storage. Approximately 48K words of core.

2.2.2.3 Run Time. Approximately 117 cp seconds and 50 minutes wall clock time.

2.2.2.4 Cost. Approximately 350 dollars.

2.2.2.5 Program Function. RSS2 converts the X-Y pairs describing each feature boundary into horizontal strip format. This is done for both "line" features such as roads and "area" features such as lakes. Clearly a point feature such as a building can be displayed by a single strip of unit length. For area features, long horizontal "fill" strips are generated so that the area enclosed by the feature boundary is now part of the feature.

2.2.2.6 Program Description. The process of strip generation takes place in three different stages. These stages are point processing, line processing, and polygon processing. Each feature is defined as either a point, line, or polygon feature. If the feature is a point feature, then a six word unit strip is formatted for each point of the feature into an output buffer in the order that they are input by RSS1. These unit strips are then ordered by Y and then X. After sorting is complete, a record header is added to the buffer (array IX) and the buffer is output to a disk file. For line processing, the following procedure takes place. For a consecutive point pair, say (X_1,Y_1) and (X_2,Y_2), ΔX and ΔY is computed. To construct the data strips corresponding to these two points begin by calculating the inverse slope.

$$S=\frac{\Delta X}{\Delta Y} = \left| X_1-X_2/Y_1-Y_2 \right|$$

This slope is then scaled to a 2048 grid. $S = 2048 * \frac{S_1}{\pi}$

If ΔY is not θ, then S becomes: $S = S \pm (2*2048)$

Note: $\pm$ depends on direction of slope

The number of strips to be generated is given by $|Y_1 - Y_2|$, each strip being of unit width.

For the first strip, the values computed are Y=Y, X=X, and $\Delta X = X+S$. For each of the remaining strips, $Y=Y_1-n$, $X=X_1+S+n$, and $\Delta X = S$ where n equals the number of strips processed for this point pair. The Y, X, and ΔX values for each strip is formatted into a work area (array JX). After the strips are generated for this point pair, the recently generated strips in the JX array are sorted into the previously generated strips of the IX array ordered by Y, X, ΔX, θ respectively. The above procedure is repeated for each point pair of the feature. Polygon processing follows line processing. If the line feature already processed is determined to be a non-polygon or filled polygon with no overlaps, the feature record with its header added is output to a disk file. Otherwise a series of tests are made for each pair of line segments (strips) in the feature to determine where gaps or overlaps occur. If a gap exists between a pair of line segments, a filler strip is formatted and inserted in the proper order into the output buffer (IX array). If an overlap exists between any two line segments, say X_1 and X_2, then X_2 is shortened to eliminate the gap.

The following is a description of the important variables in RSS2.

Label	Type	Description
CONV	Real	Scaling factor to convert X-Y points to grid resolution units.

Label	Type	Description
DISK	Integer	Variable which defines the strip feature output device unit. Disk = 9
DX	Real	Δ X for any consecutive point pair.
DY	Real	Δ Y for any consecutive point pair
IDDDX	Integer	Δ X value, may be positive or negative
IDDX	Integer	X displacement of line segment from vertex.
IDX, IDY	Integer	Integer representations of DX, DY
IF	Integer	Feature code of current feature
IFG	Integer	Flag indicating direction of line segment. Negative value means left edge, positive value means right edge, zero means horizontal.
IFILL	Integer	Counter flag indicating whether fill strip is required.
IGDSIZ	Integer	Grid resolution size in meters X1000
IIF	Integer	Merge priority code for feature
IIT	Integer	Number of segments in the JX+IX arrays
IN	Integer	Number of coordinates in feature record
INTAP	Integer	Variable which defines the input device unit number. INTAP=8
IN2W	Integer	Variable which defines the output device unit number for 2 word buffer. IN2W=20
IRATN	Integer	Length of line segment required
IREGSZ	Integer	Region size on a side
IREM	Integer	Same as ISEG
IRES	Integer	Grid points per line. IRES=2048
IRTND	Integer	X displacement of line segment from vertex
ISEG	Integer	Number of line segments in feature record

ISEQ	Integer	Sequence number of feature
ISET	Integer	Map segment counter
ISPEC	Integer	Specularity flag
IT	Integer	Hold area for ISEG
ITH	Integer	Grid scaled theta value
ITRC	Integer	Counter of attempted reads
ITYP	Integer	Feature type. 1=point, 2=line, 3=polygon
IX	Integer Array	Feature output buffer
JSEG	Integer	Number of line segments in JX array
JT	Integer	Hold area for JSEG
JX	Integer Array	Array of line segments for current strip
KX	Integer	Array of grid scaled data points
MWD	Integer	Number of words read for this feature
NI	Integer	Top of IX list
NII	Integer	Number of entries in the IX+JX buffers
NIJ	Integer	Pointer to second element of IX list
NJ	Integer	Top of JX list
NK	Integer	Position pointer for fill strip into IX buffer
NSEG	Integer	Number of line segments required to bridge current point pair
NWD	Integer	Length of feature record
PI	Real	Inverse tangent of one unit
RATIO	Real	Absolute value of $\Delta X/\Delta Y$

RATN	Real	Value represents X+(ΔX/ΔY)
RESK	Real	Resolution element size in feet
SCALE	Real	Scale factor. (ex. 100000)
TH	Real	Theta value unscaled to grid
WORD	Real Array	Array for 2 word output buffer
X2	Real Array	Input feature buffer
X5	Integer Array	Table for conversion of feature codes merge priority codes

2.2.2.7 Input. RSS2 requires one disk input file (Tape 8) from RSS1 and one card input file with the following format.

Columns 1-20	Contains the conversion factor for changing inches on the map to feet on the ground. For example, if the map scale is 1:1000,000, this field contains the number 8333.333
Columns 21-35	Blank
Columns 36-40	Contain the size of the grid resolution elements used to label points in a region of planimetry data. This value must be expressed in units of meters X 1000.
Columns 41-48	Blank
Columns 49-50	Contain the region size, i.e. the number of resolution elements along a region edge. The restrictions on this value are (1) it must not exceed 48 because RSS7 which processes by region has core allocation for a maximum 38 x 48 region size; and (2) it must be a multiple of 4 because the input data for RSS7 is packed four data items per word and this requirement would eliminate any need for unpacking and repacking of data.

2.2.2.8 Output. RSS2 outputs two disk files. The first output file (Tape 9) contains 1800 word feature records. Each record has a 6-word header with the remaining record divided into 4-word strip descriptions. The format of the feature buffer is as follows.

Header - Words 1-6

Word 1	Feature sequence number
Word 2	Feature code
Word 3	Number of coordinates in feature buffer
Word 4	Number of line segments in feature buffer
Word 5	Specularity flag for feature
Word 6	Feature type 1=point, 2=line, 3=polygon

Strip Description - Word 6-1800

Word 6+4N-3	Y-value for the Nth strip
Word 6+4N-2	X-value for the beginning of the Nth strip
Word 6-4N-1	ΔX-value for the Nth strip, equals the total number of grid points covered by the strip. The endpoint of the strip is then set to X+ X-1.
Word 6+4N	Specularity angle for the Nth strip. This quantity is used to determine abnormal radar return quantities. Presently not used.

The second output file (Tape 20) consists of a single two-word entry.

Word 1	Resolution element size in meters X 1000
Word 2	Number of resolution elements along a region edge

This file is created on disk to eliminate the need for card input of the same data throughout the remainder of the system.

2.2.2.9 Externals. System routines called by RSS2 are LENGTH and UNIT which are mass storage I/O function routines, and ATAN2 which computes the inverse tangent between any two arguments. There are no subroutine calls in this program.

2.2.2.10 Error Conditions. RSS2 has three error conditions. One error condition occurs if the card input resolution size or region size is out of the allowable range. The error messages output for this condition are as follows:

INPUT ERROR-RESOLUTION ELEMENT TOO FINE
OR
INPUT ERROR-REGION SIZE TOO LARGE

Another error condition occurs if there is an error found while writing either of the two output buffers of this program. The error message output is as follows:

OUTPUT ERROR ON UNIT nn

The next error condition occurs if an error is found while reading the feature input buffer. The message for this condition is as follows:

IN FEATURE NO. nnnn ERROR AT Y=nnnn

2.2.2.11 Program Flowchart. The flow diagram for RSS2 is illustrated in Figure II.4

2.2.2.12 Program Listing. The program listing for RSS2 is attached.

2.2.3 and 2.2.4

2.2.3.1 Program Name. RSS3

2.2.3.2 Storage. Approximately 21K words of core

2.2.3.3 Run Time. Approximately 230 cp seconds and 2 minutes wall clock time.

2.2.3.4 Cost. Approximately 35 dollars

2.2.3.5 Program Function. RSS3 assigns the planimetry data strips to regions. If a strip overlaps several regions, say N of them, it is broken up into N segments with each segment being assigned to its respective region. RSS3 also assigns merge priorities for each feature code.

2.2.3.6 Program Description. The procedure used to assign the planimetry strips to their respective regions is a simple one. Consider for example a strip with the following characteristics:

$$Y = 750$$
$$X = 875$$
$$\Delta X = 24$$
Region Size = 32
Resolution = 156.25 feet

This means that the strip begins 117,187 feet north (750 X 156.25) and 136,718 feet east (875 X 156.25) of the map origin. We have assumed that each region contains 32 increments along each edge so:

$$R_Y = 750/32 = 23.43750$$

$$R_x = 875/32 = 27.34375$$

If we assume that each horizontal row of regions can contain at most 331 regions, then the region number for the beginning of the strip is:

$$N_r = 331 * (23) + 28 = 7641$$

The (X,Y) coordinates of the beginning point relative to the first grid element in the region is:

$$Y = 0.43750 * (32) = 14$$

$$X = 0.34375 * (32) = 11$$

Since $X = 24$, the end point of the strip is at $X = 11+24-1 = 34$. The maximum address within a region is 32, so this strip overflows into the next region to the east. We therefore wind up with two strips as follows:

	STRIP 1		STRIP 2
Region	7641	Region	7642
Y	14		14
X	11		1
Δ X	22		2

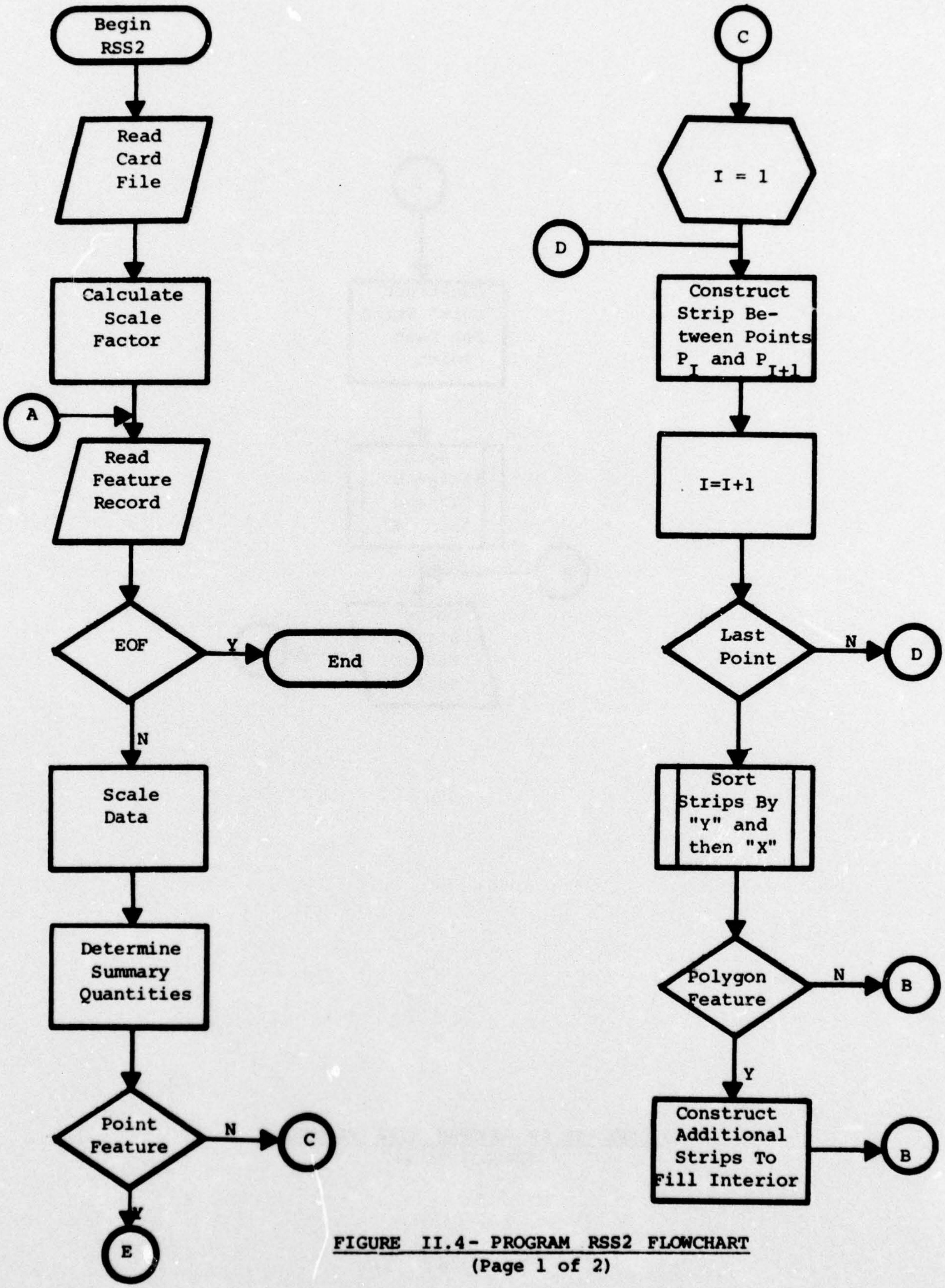


FIGURE II.4- PROGRAM RSS2 FLOWCHART
(Page 1 of 2)

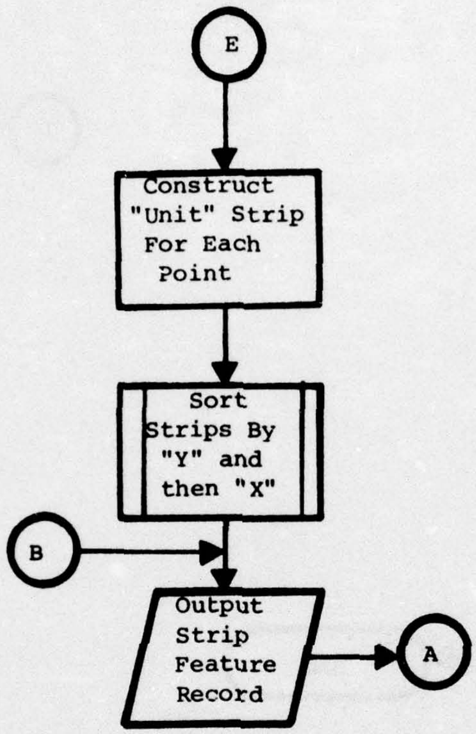


FIGURE II.4- PROGRAM RSS2 FLOWCHART
(Page 2 of 2)

Numbering of these regions begins in the southwest corner of the map with the assignments being made sequentially left to right and bottom to top. The merge priorities assigned to each feature are done through the use of a translation table which is hard-coded as a DATA statement.

The following is a description of the important variables in RSS3.

Label	Type	Description
DISK	Integer	Variable which defines the input unit device number for the planimetry strip feature data. DISK=7
IF	Integer	Feature code
IFC	Integer Array	Merge priority conversion table
II	Integer	Y value for the southwest corner of the region
III	Integer	Region count in Y direction
ILIM	Integer	Pointer to segment count in feature record.
IMAX	Integer	Maximum X value for Y
IMIN	Integer	Minimum X value for Y
IN	Integer Array	1800 word input buffer for strip feature record
INRC	Integer	Number of records output
INSG	Integer	Number of segments output
IN2W	Integer	Variable which defines the input device number for the 2-word file from RSS2. IN2W=20
IREG	Integer	Region number
IRES	Integer	Number of points per plot line
ISEG	Integer	Number of segments in record

ISEQ	Integer	Sequence number
ISPEC	Integer	Specular code
IST	Integer	Counter for number of words of feature data input
ITYP	Integer	Feature type
IXL1	Integer	Region number for IMIN
IXL2	Integer	Region number for IMAX
IYL1	Integer	Y value for bottom of feature
IYL2	Integer	Y value for top of feature
JDX	Integer Array	Array of ΔX values for disk output
JJ	Integer	X value for the southwest corner of the region
JJJ	Integer	Region count in X direction
JTH	Integer Array	Array for θ values for disk output
JX	Integer Array	Array for X values for disk output
JY	Integer Array	Array for Y values for disk output
MWD	Integer	Number of words transferred from input buffer
NRX	Integer	Number of regions in the X direction
NVM	Integer	Number of segments in record
NX	Integer	Number of elements in X direction of region
NY	Integer	Number of elements in Y direction of region
OUTAP	Integer	Variable which defines output unit device number. OUTAP=1
WORD	Integer Array	Array for 2-word buffer output by RLMS2

2.2.3.7 Input. RSS3 requires two disk input files. The files required are the two word output file (Tape 20) from RSS2, and the strip feature data file (Tape 7) from RSS2.

2.2.3.8 Output. RSS3 outputs one disk file (Tape 2) in the form of card images. The format for each image is as follows:

Word 1-5	Region number
Word 6-7	Merge priority code
Word 8-9	Feature code
Word 10	Feature type
Word 11	Specularity code
Word 12	Number of segments in record
Word 13-89	Y, X, ΔX, θ for up to seven segments

The only printout from RSS3 is a message indicating the end of processing.

2.2.3.9 Externals. System routines called by RSS3 are LENGTH and UNIT which are mass storage I/O function routines. There are no subroutine calls in this program.

2.2.3.10 Error Conditions. RSS3 has two error conditions. One error condition occurs when there is an error found while reading the two word disk file output by RSS2. The error message for this condition is as follows:

INPUT ERROR ON UNIT 20

The other error condition occurs when an error is found while reading the input strip feature file output from RSS2. The error message for this condition is as follows:

BUFFER IN ERROR AT ISEQ = XXXXXXX

2.2.3.11 Program Flowchart. The flow diagram for RSS3 is illustrated in Figure II. 5

2.2.3.12 Program Listing. The program listing for RSS3 is attached.

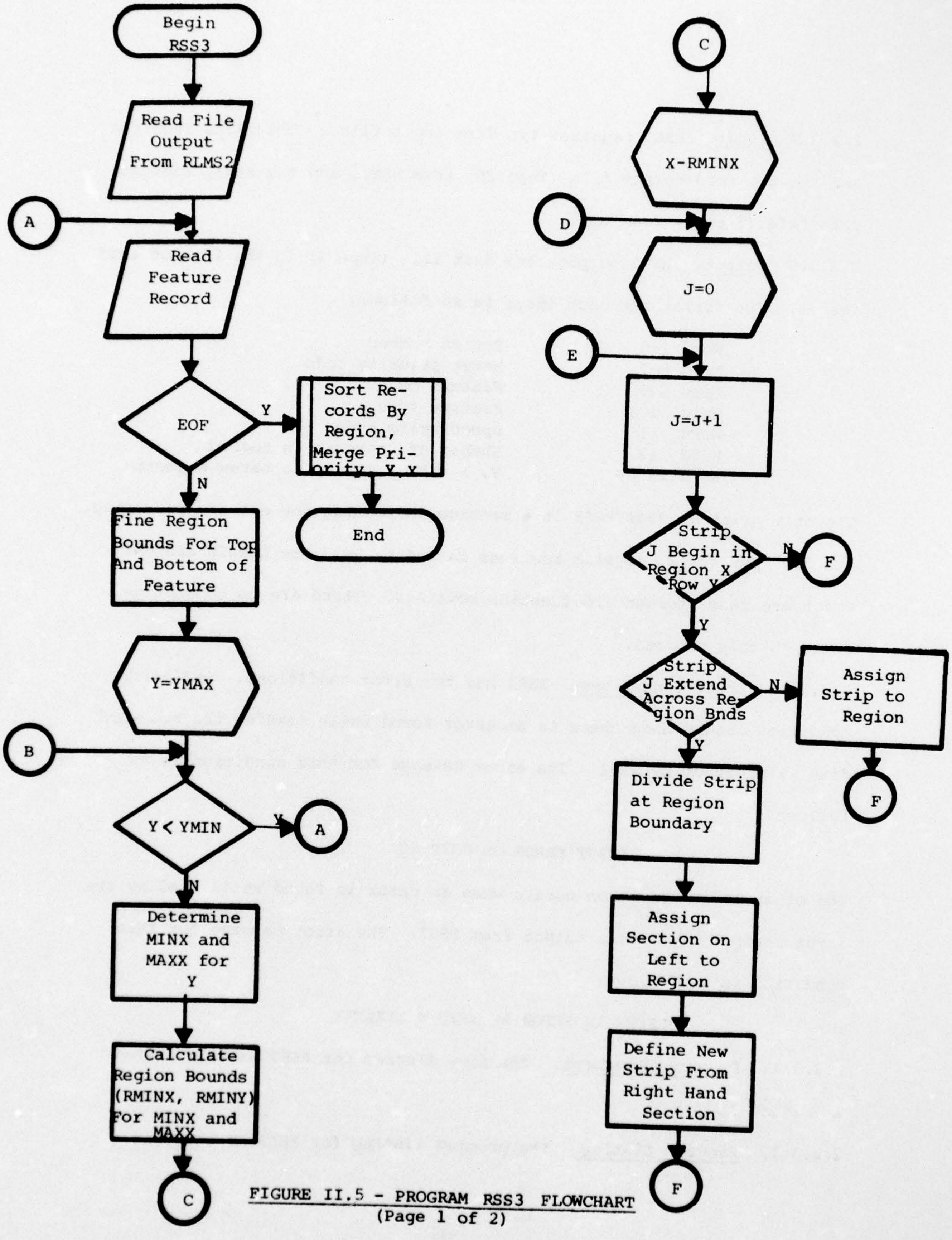


FIGURE II.5 - PROGRAM RSS3 FLOWCHART
(Page 1 of 2)

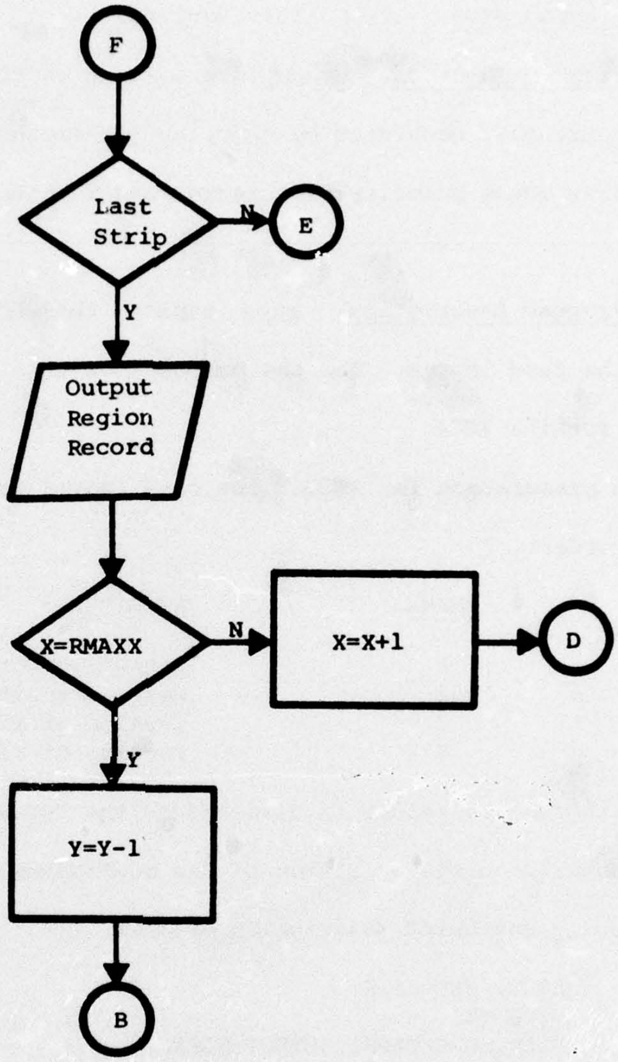


FIGURE II.5 - PROGRAM RSS3 FLOWCHART
(Page 2 of 2)

2.2.4.1 Program Name. RSS4 (Sort Routine)

2.2.4.2 Program Function. RSS4 is a sorting routine which reorders the planimetry data generated by RSS3 by (1) ascending region number; (2) ascending merge priority; (3) increasing Y value; and (4) increasing X value.

2.2.4.3 Program Description. RSS3 employs the CDC SORT/MERGE package to order the card images. For the purposes of this document, we will call this routine RSS4.

In preparation for RSS5, the card images are sorted in the following order:

ORDER	ATTRIBUTE
1	Region Number
2	Merge Priority
3	Y-value of first strip
4	X-value of first strip

Although the reader is directed to the CDC SORT/MERGE manual for complete details on the operation of the SORT/MERGE package, we present here a listing and brief description of RSS4.

```
SORT, VAR=DISK
BYTESIZE,6
FILE,SORT=TAPE1,OUTPUT=TAPE2
FIELD,REGION(1.1,5,DISPLAY),PRIORITY(6.1,2,DISPLAY),
  ,INTY (13.1,2,DISPLAY) ,INTX(15.1,2,DISPLAY)
KEY,REGION(A,OWN) ,PRIORITY (A,OWN),INTY(A,OWN),INTX(Z,OWN)
SEQUENCE,OWN(b,0,1,2,3,4,5,6,7,8,9)
OPTIONS,RETAIN
END
```

The FIELD card lists the attributes to be sorted in order of decreasing priority. It indicates the format of the data, which in our case is CDC DISPLAY code, and its location within the record. The KEY

card gives the direction of the sort (i.e. the "A" indicates that the numbers are to be placed in ascending order) and the ordering scheme, which in our case is the sequence OWN. The SEQUENCE card is used to define an ordering sequence other than one of the standard sequences. This card is required in RSS4 because a blank in display code is represented by 55B and this value is greater than that assigned to the integers. Without a special sequence, a number like ƀƀ1023 would, for example, be placed after ƀ16486 in an ascending order sort of the card images.

Clearly, the output from RSS4 is in the same format as that from RSS3. Only the ordering of the records is changed.

2.2.4.4 Input. RSS4 requires the disk output planimetry data file (Tape 2) from RSS3.

2.2.4.5 Output. The disk output from RLMS4 is the same format as that from RSS3. Only the ordering of the records has changed.

2.2.5

2.2.5.1 Program Name RSS5

2.2.5.2 Storage Approximately 44K words of core

2.2.5.3 Run Time Approximately 217 cp seconds and 2 minutes wall clock time

2.2.5.4 Cost. Approximately 45 dollars

2.2.5.5 Program Function. RSS5 merges the planimetry strips so that no strips overlap. It marks the end of the planimetry data preparation.

2.2.5.6 Program Description. The records containing the planimetry strips have already been sorted by region number and merge priority. The merge priority scheme is set up so that features which may be overwritten appear before those which may not. For example, consider two features, a city (low merge priority) and a lake (higher merge priority). Each of these has associated with it a series of strips describing its shape and location.

Now, if the lake lies within the bounds of the city, a merge of the data is required since it is necessary to specify which feature is to be "on top"; i.e. the lake strips must be copied over the city strips or the lake will not appear on the final picture. The scheme is presently imperfect (consider an island with a lake on it) but can be expected to work in the vast majority of cases.

In essence, RSS5 simply copies the planimetry strips into a core array in the order in which they appear on the input file. This array becomes an image of the region structure as successive strips are copied into it. An empty record is written for those regions without planimetry data. As an example of the operation of RSS5, consider two planimetry strips, one being part of a lake and the other being part of a city. Take the location and size of these strips to be as follows:

Lake Strip	City Strip
Y = 25	Y = 25
X = 11	X = 8
ΔX = 6	ΔX = 15

Clearly, this portion of the lake overlaps the city. RSS5 takes these strips and generates three (3) strips, two belonging to the city and one belonging to the lake. The location and size of these strips is as follows:

Lake Strip	City Strip	
Y = 25	Y = 25	Y = 25
X = 11	X = 8	X = 17
ΔX = 6	ΔX = 3	ΔX = 6

Notice that these strips do not overlap.

The following is a description of the important variables in RSS5.

Label	Type	Description
DISK	Integer	Variable which defines the input unit device number for the planimetry data. DISK=7
IEND	Integer	End of file flag
IF	Integer Array	Common block of feature codes for first or previous strip
IG	Integer Array	Common block of merge priority codes for first or previous strip
IKK	Integer	Merge priority for input strip
IKC	Integer	Region size of planimetry data on a side
IKC2	Integer	Total number of elements in a region
IN	Integer	Number of segments input
INDEX	Integer Array	Storage area required by mass storage I/O routines
IN2W	Integer	Variable which defines the input device number for the 2 word file from RSS2 IN2W=20
IREG	Integer	Input region number
ISUM	Integer	Total number of elements in the current region
ITH	Integer Array	Common block of specularity angles for first or previous strip
IX	Integer Array	Common block of Y-values for first or previous strip
IY	Integer Array	Common block of X-values for first or previous strip
IZ	Integer Array	Common block of ΔX values for first or previous strip

JF	Integer Array	Common block of feature codes for current strip
JG	Integer Array	Common block of merge priority codes for current strip
JN	Integer	Number of segments in current strip
JREG	Integer	Current region number
JTH	Integer Array	Common block of specularity angles for current strip
JX	Integer Array	Common block of Y values for current strip
JY	Integer Array	Common block of X values for current strip
JZ	Integer Array	Common block of ΔX values for current strip
KF	Integer Array	Common block work area for merged feature codes
KG	Integer Array	Common block work area for merged priority codes
KTH	Integer Array	Common block work area for merged specularity angles
KX	Integer Array	Common block work area for merged Y values
KY	Integer Array	Common block work area for merged X values
KZ	Integer Array	Common block work area for merged ΔX values
N,K	Integer	Number of elements (words) in region
NBYT	Integer	Counter for number of elements output
NEMT	Integer	Counter for number of empty regions output
NFUL	Integer	Counter for number of filled regions output
NSEG	Integer	Counter, not used

NTOT	Integer	Counter not used
NY	Integer	Maximum number of regions in Y direction
NX	Integer	Maximum number of regions in X direction
WORD	Integer	Array for two word buffer output by RLMS2

2.2.5.7. Input RSS5 requires two input disc files. The files required are the two word output file (TAPE 20) from RSS2, and the sorted output file (TAPE 7) from RSS4.

2.2.5.8 Output. RSS5 outputs one random access disc file (TAPE 10). This output file consists of a record for each region on the map regardless of whether or not that region actually contains any planimetry data. The first word of the record contains the total number of strips contained in the region, while the remaining words contain the information describing the strips. Each strip is described by one word, with the data being packed as follows:

BITS	INFORMATION
31 - 36	Y - coordinate of strip
25 - 30	X - coordinate of strip
19 - 24	ΔX for strip
7 - 18	Specularity angle for strip
1 - 6	Feature code for strip

The only printout from RSS5 is a message indicating the end of processing.

2.2.5.9 Externals. System routines called by RSS5 are OPENMS, CLOSMS, WRITMS, and UNIT which are mass storage I/O function routines. The subroutine called by RSS5 is RADOUT. This subroutine is discussed under section 2.2.5.13.

2.2.5.10 <u>Error Conditions.</u> RSS5 has three error conditions. One error condition occurs if there is an error found while reading the two word disc file output by RSS2. The error message for this condition is as follows:

INPUT ERROR ON UNIT 20

Another error condition occurs if the total number of elements ISUM output by a region is not equal to the square of the region size on a side (IKC2). If ISUM is larger than IKC2, then the 500 word common blocks will overflow. If ISUM is smaller than IKC2, part of the region record will be empty. The error message output for this condition is as follows:

PARITY ERROR DURING INDEX WRITE-OUT

2.2.5.11 <u>Program Flowchart</u>. The flow diagram for RSS5 is illustrated in Figure II.6.

2.2.5.12 <u>Program Listing</u>. The program listing for RSS5 is attached.

2.2.5.13 <u>Subroutine Description.</u>

2.2.5.13.1 <u>Subroutine Name</u>. RADOUT

2.2.5.13.2 <u>Summary</u>. RADOUT packs and writes out the planimetry region records.

2.2.5.13.3 <u>Description of Processing</u>. The calling sequence for RADOUT is - RADOUT(K). K is the number of features in the region. For each feature, pertinent information is packed in to one word describing that feature and stored into a region record. When the region record packing is complete, the record is output to a random access file. For a

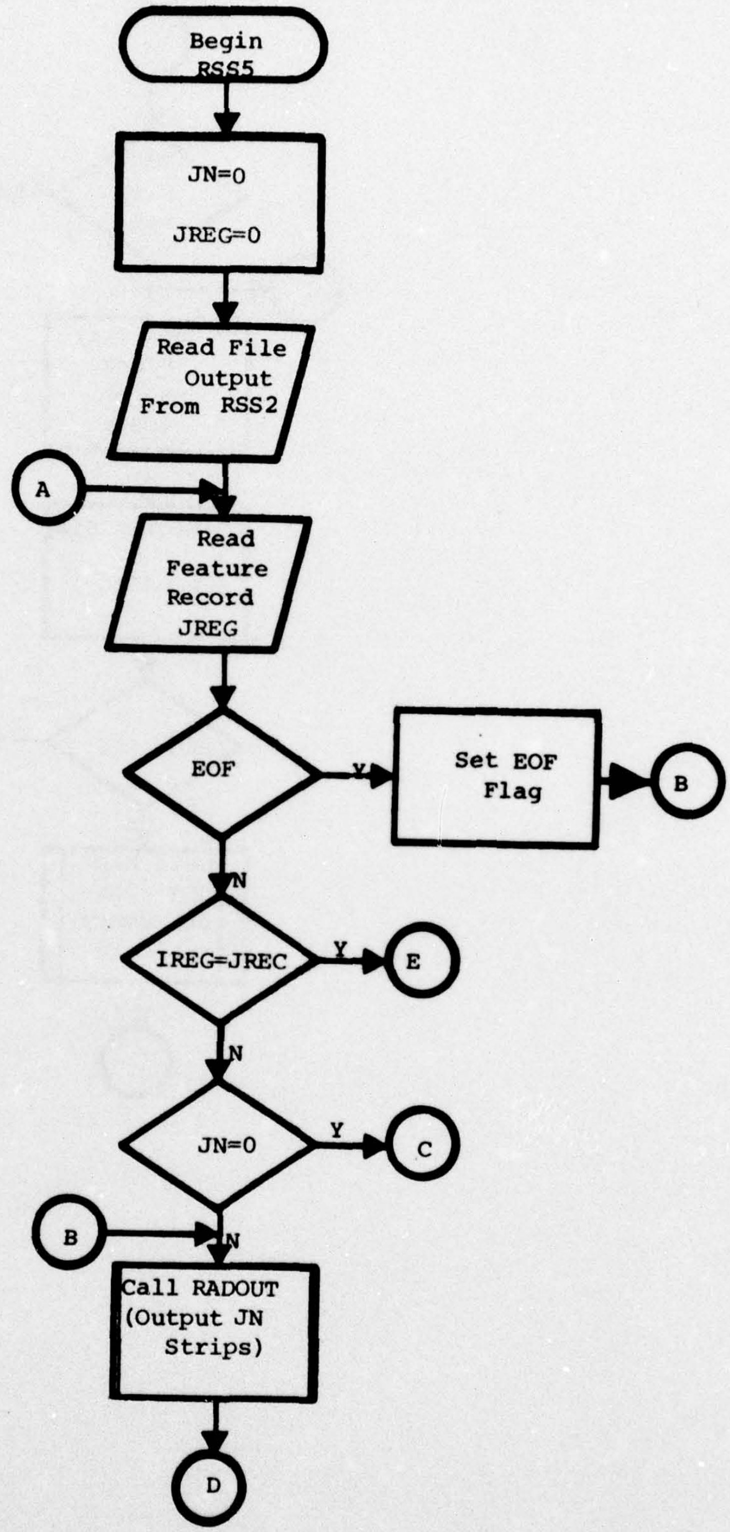


FIGURE II.6 - PROGRAM RSS5 FLOWCHART
(Page 1 of 4)

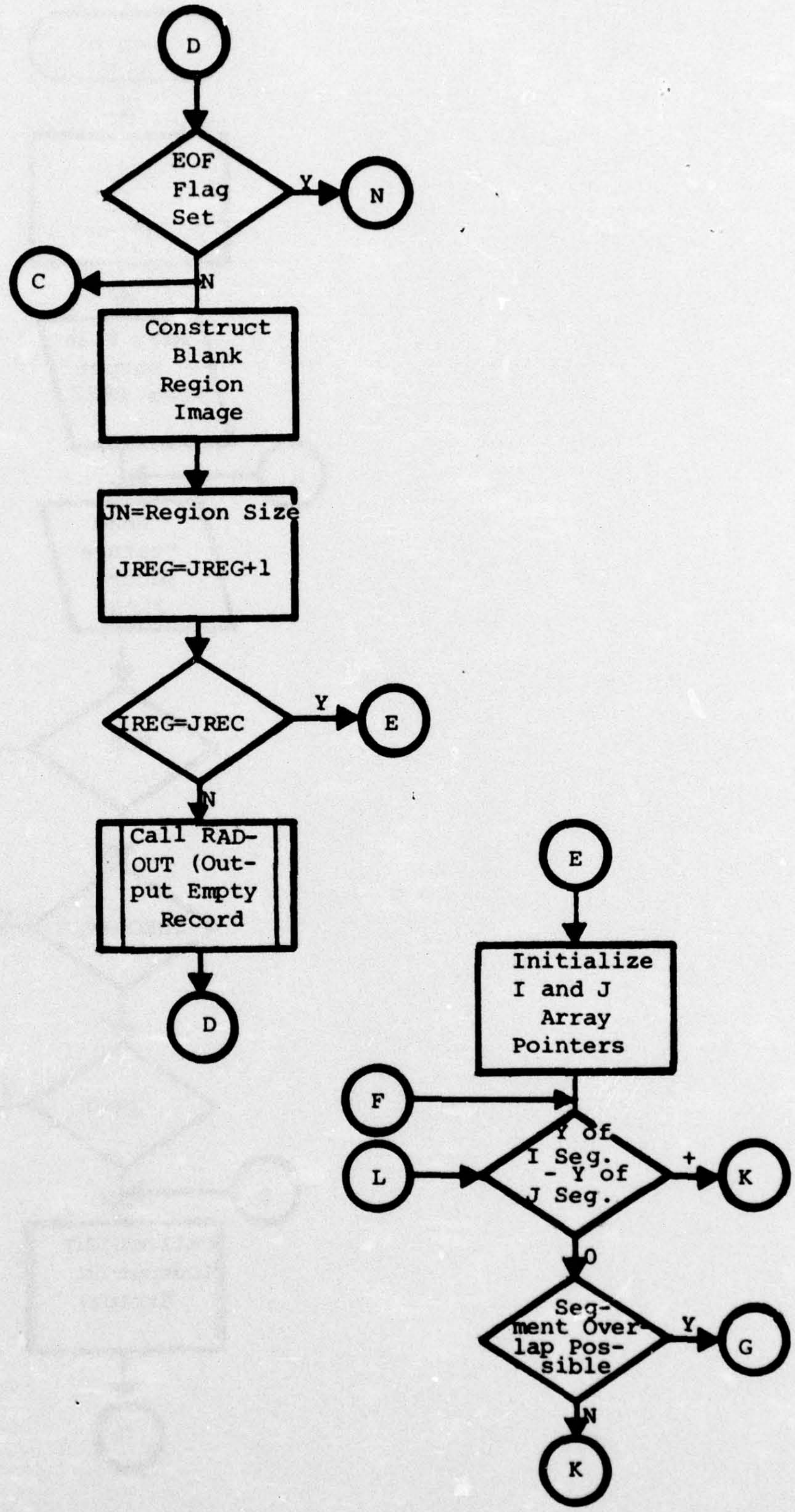


FIGURE II.6 - PROGRAM RSS5 FLOWCHART
(Page 2 of 4)

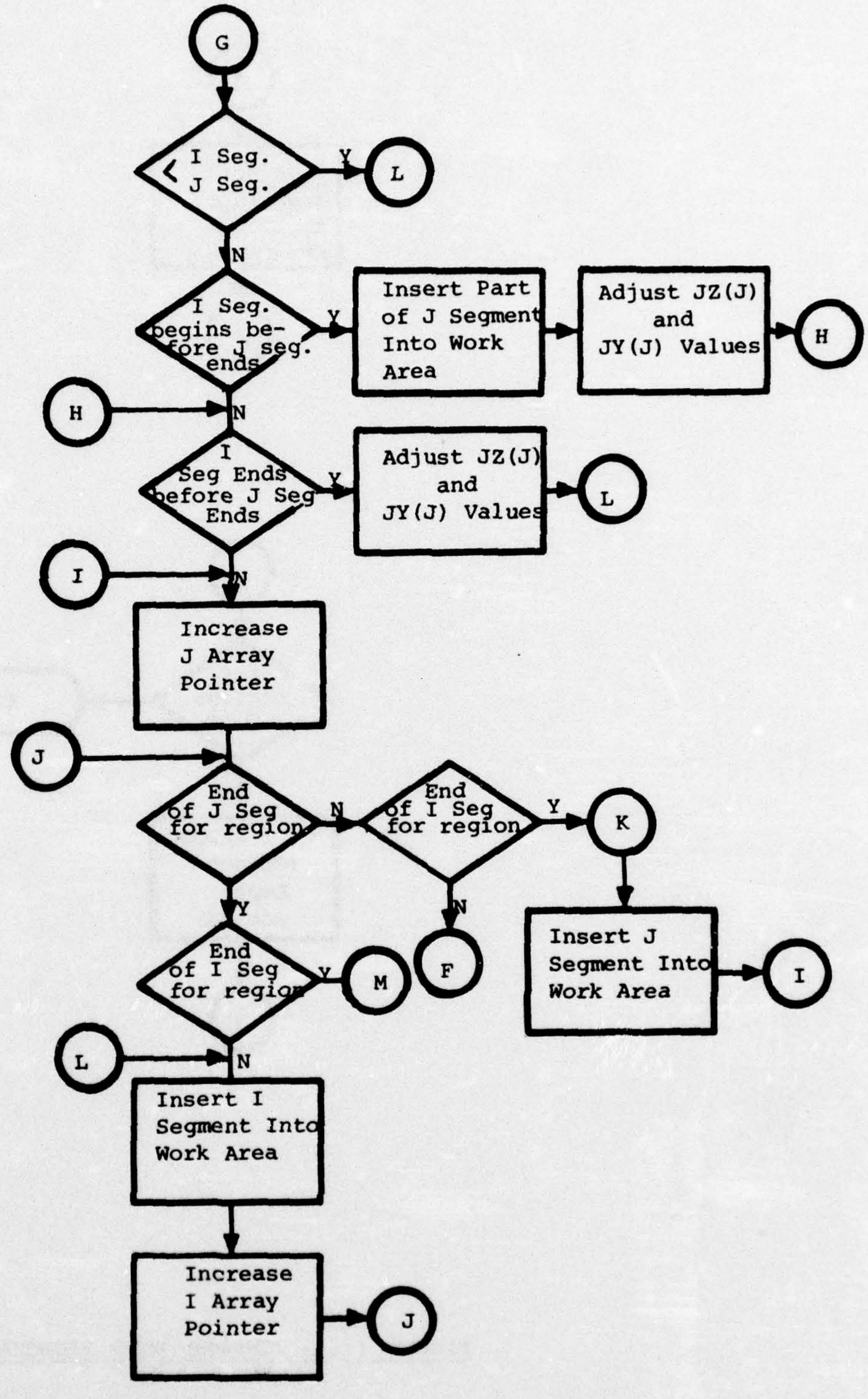


FIGURE II.6- PROGRAM RSS5 FLOWCHART
(Page 3 of 4)

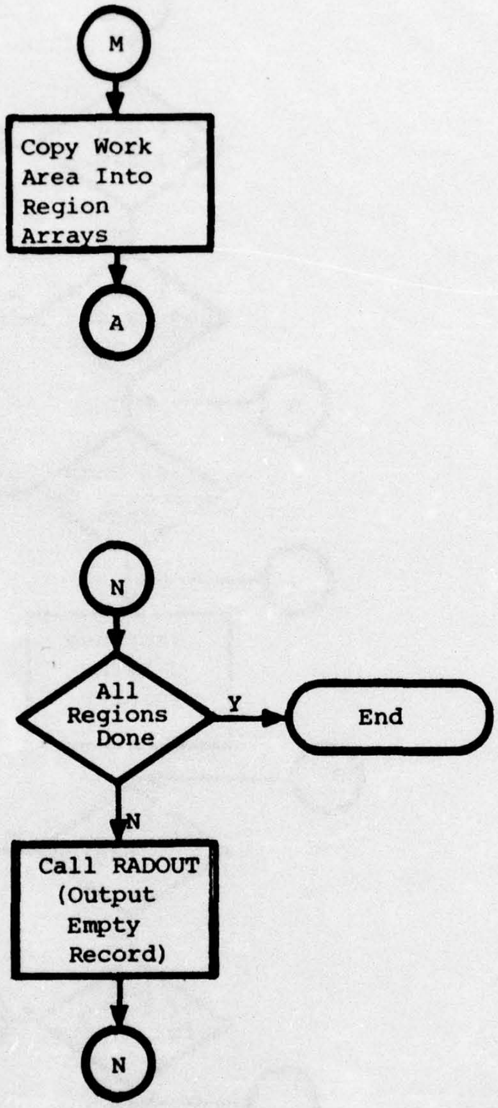


FIGURE II.6 - PROGRAM RSS5 FLOWCHART
(Page 4 of 4)

description of the important variables in this subroutine refer to the list under section 2.2.5.6.

2.2.5.13.4 Error Conditions. None

2.2.5.13.5 Subroutine Flowchart. A flow diagram is not considered necessary. Consult the listing of RSS5 for details.

2.2.5.13.6 Subroutine Listing. The listing for RADOUT is included with that of RSS5.

2.2.6

2.2.6.1 Program Name. RSS6

2.2.6.2 Storage. Approximately 64K words of core.

2.2.6.3 Run Time. Approximately 117 cp seconds and 1 minute wall clock time.

2.2.6.4 Cost. Approximately 43 dollars.

2.2.6.5 Program Function. RSS6 divides the terrain data into region blocks.

2.2.6.6 Program Description. The input data base is characterized by resolution elements whose length is defined by word 34 of the header record, in units of meters x 1000. The resolution elements are arranged in horizontal strips covering the entire width (east-west dimension) of the map sheet. Word two of a two word disc input file defines the region size in resolution elements or a side. Horizontal strips of data whose length is defined by the region size is read into core. This data is reformatted into rows of region blocks numbered from the lower left corner of the map. Region numbers are determined by words 21 and 22 of the header record. Word 21

defines the number of regions in the Y direction and word 22 defines the number of regions in the X direction. After all the regions of the map formatted and output to a temporary random disk file, the region records are reread in order and output to a permanent sequential disk file.

The following is a description of the important variable for RLMS6.

LABEL	TYPE	DESCRIPTION
ADDR	Integer	Pointer to starting address of data on mass storage device.
IARRAY	Integer Array	Temporary holding area of data for one region.
IK	Integer	Region size of data (in meters x 1000)
IM	Integer	Number of words in a region record.
IN	Integer	Number of words required to read in data for one row of regions.
INBUF	Integer Array	Input buffer for terrain tape data.
INDEX	Integer Array	Storage area required by the mass storage I/O routines.
INTAP	Integer	Variable which defines the input unit device number for the terrain data. INTAP = 8
INZW	Integer	Variable which defines the input device number for the 2 word file from RSS2. INZW = 20.
IPOINT	Integer	Pointer to data item within current row of regions.
JJ	Integer	Region row number
K	Integer	Number of words currently in region record currently being formatted.

MOVE15	Integer Array	Array defining the number of bytes to shift a word of the packed input record.
NCOL	Integer	Number of data elements in the X direction.
NREC	Integer	Number of records
NREGION	Integer	Region number
NROW	Integer	Number of data elements in the Y direction.
NXREG	Integer	Number of regions in the X direction.
NYREG	Integer	Number of regions in the Y direction.
OUT	Integer Array	Buffer for formatted output region record.
OUTAP	Integer	Variable which defines output device unit number for formatted region record. OUTAP = 9
WORD	Integer Array	Array for two word buffer output by RSS2.
XCOOR	Integer	Current X coordinate position.
XFILL	Integer	Number of filler points required, if any, in the X direction.
YCOOR	Integer	Current Y coordinate position.
YFILL	Integer	Number of filler points required, if any, in the Y direction.

2.2.6.7 Input. RSS6 requires one input tape file and one input disk file. The tape file (TAPE 8) contains the terrain data base. The disk file (TAPE 20) contains the two word buffer output by RSS2.

The first record on the tape is a 36 word header. Each entry in the header is an integer representation of the actual valve multiplied by 1000 and truncated. The following is the description of the header.

Word 21: The total number of records (profiles) on the tape, not counting the header.

Word 22: The total number of elevation points per record. Elevations are unsigned 15 bit integers, in feet, packed 4 per word.

Word 23-25:

Geographic latitude in degrees, minutes, and seconds of the radar target.

Word 26: Ground distance in meters between the UNAMACE point of tangency and the radar target.

Word 27-29:

The longitude in degrees, minutes, and seconds of the radar target.

Word 30: The ground distance in meters from the UNAMACE point of tangency west to the target.

Word 31: The ground distance in meters from the first elevation profile (upper NW corner of the elevation data) north to the radar target. A negative value indicates that the target is south of the first elevation point.

Word 32: The ground distance in meters from the first elevation profile west to the target. This number is also negative.

Word 33: Target ID Code

Word 34: Ground spacing between points of a profile in meters. Presently this value must be equal to the grid resolution size output by RSS2.

Word 35: Ground spacing between profiles in meters. Equals word 34 at present time.

Word 36: Zero fill word.

Following the header are a series of records - one for each profile line. Each profile contains the elevation values for a west-to-east running strip of terrain, with the first profile corresponding to the northernmost part of the map (a contrast to the SW origin for digitizing the planimetry).

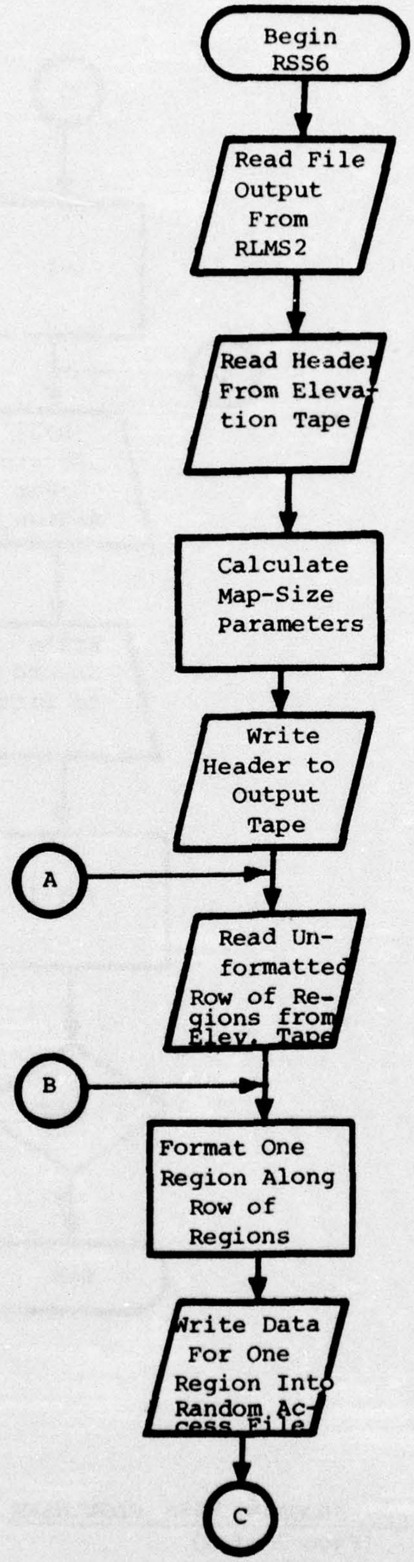


FIGURE II.7 - PROGRAM RSS6 FLOWCHART
(Page 1 of 2)

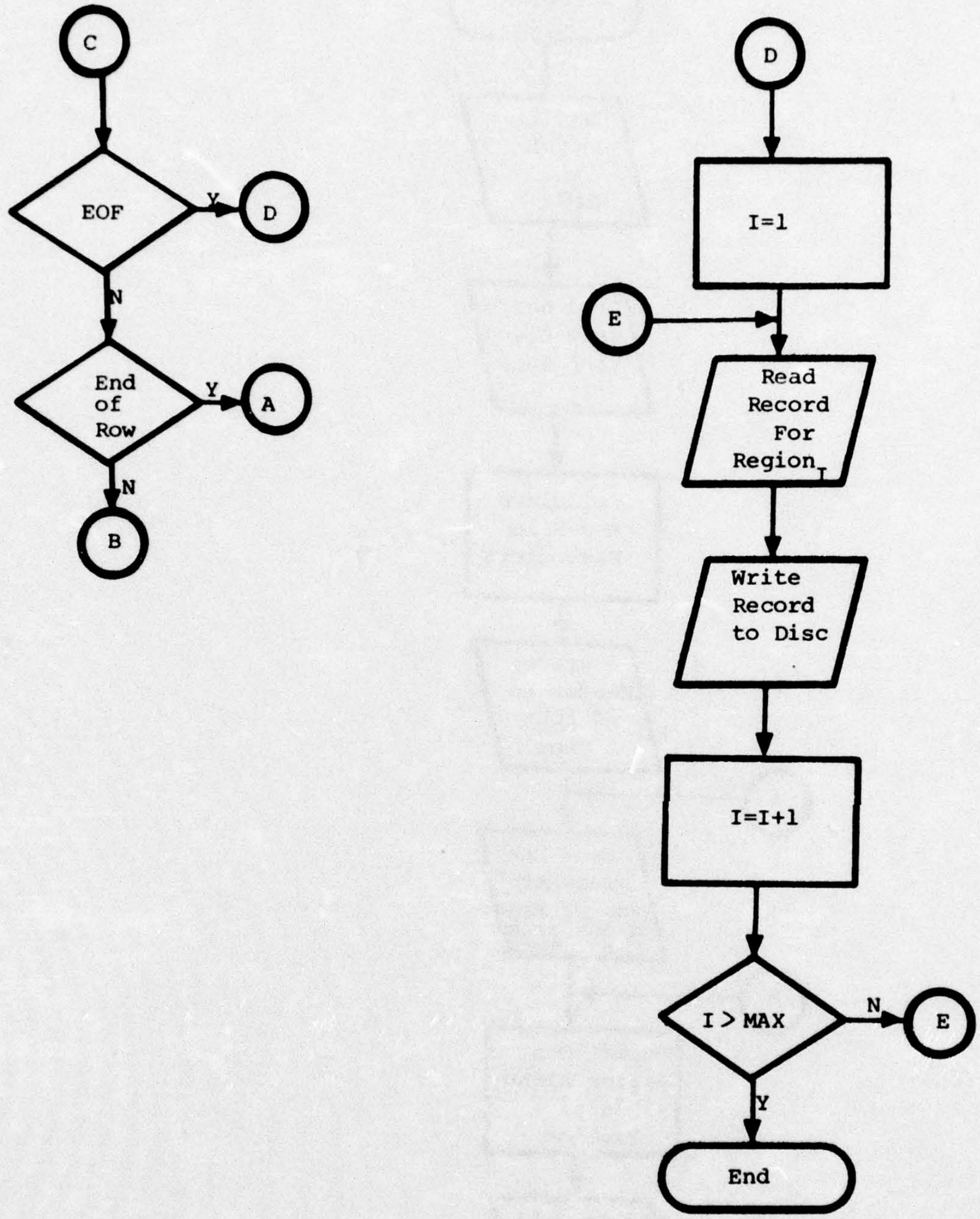


FIGURE II.7 - PROGRAM RSS6 FLOWCHART
(Page 2 of 2)

2.2.6.8 Output RSS6 has one disk output file (Tape 9) arranged such that the information for each region is stored in a one-dimensional array. The first word contains the region number while the remaining words contain the elevation data packed four per word. The data is arranged such that the southwestern edge of the map appears first. The only printout from RSS6 is a message indicating the end of processing.

2.2.6.9 Externals System routines called by RSS6 are OPENMS, READMS, WRITMS, and UNIT which are mass storage I/O function routines. There are no subroutine calls made by this program.

2.2.6.10 Error Conditions. RSS6 has two error conditions. The first error condition occurs if there is an error during input or output to the disk files. One of the following messages is output for this error condition:

PREMATURE EOF ENCOUNTERED ON INPUT TAPE

OR

PARITY ERROR ENCOUNTERED......ERROR EXIT

The second error condition occurs if the resolution value for the terrain data is different from that for the planimetry data. The message output for this condition is as follows:

ERROR-CULTURE RESOLUTION NOT EQUAL TO TERRAIN RESOLUTION

2.2.6.11 Program Flowchart. The flow diagram for RSS6 is illustrated in Figure II. 7.

2.2.6.12 Program Listing. The program listing for RSS6 is attached.

2.2.7

2.2.7.1 Program Name. RSS7

2.2.7.2 Storage. Approximately 42K words of core

2.2.7.3 <u>Run Time</u>. Approximately 36 cp seconds and 2 minutes wall clock time.

2.2.7.4 <u>Cost</u>. Approximately 11 dollars.

2.2.7.5 <u>Program Function</u>. Program RSS7 merges the planimetry file from RSS5 and terrain file from RSS6 for all regions lying within the radar ground range of the target.

2.2.7.6 <u>Program Description</u>. Using information from the terrain file header record and the card input record, the region number and X-Y coordinates of the radar target are determined. The target location coordinate values are relative to the southwest corner. With the target point as its center, an imaginary circle is then constructed whose radius is equal the ground range of the radar. A sqaure circumscribed about this circle will then contain all of the regions required to produce the radar scene at the chosen altitude; the rest may be discarded. This procedure is illustrated in Figure II. 10. For each region overlapped by the circumscribed square, beginning at the southwest corner, the terrain and planimetry data are merged into a single record labeled by the region number and output to a disk file. Before termination a parameter file with related map quantities needed to construct the radar scene is set up and output to a disk file. The list of important variables for this program is as follows:

<u>Label</u>	<u>Type</u>	<u>Description</u>
ALTTDE	Real	The height in feet of the radar above the target. Maximum value allowed is 32000. Default value is 32000. This variable is part of the parameter list.

Label	Type	Description
DEPNGL	Real	The radar depression angle function which is equal to 1.5 + cot 25^{o} = 3.6445.
DATA	Integer Array	Array for 500 word input culture record and variable length output merged record.
DLINK	Real Array	Array for parameter output record of Real data values.
ICULT	Integer	Switch for culture data. 1= culture data, o=no culture data. Default is one.
IK	Integer	Region size. This variable is set equal to the value input in RSS2. This variable must be a multiple of 4 and have a maximum value of 48. This is part of the parameter list.
IFREQ	Integer	The number of radial scan lines to be constructed per degree. This variable is part of the parameter list.
ILIM, JLIM	Integer	Computer column and row of upper corner region of radar range.
IM	Integer	Number of words in terrain data input record.
INDEX, IDEXM1	Integer	Pointers into LDATA array
IN2W	Integer	Input buffer for two word file output by RSS2.
IPNT	Integer	Pointer used to isolate the target altitude value in the packed output buffer.
IREG, REG	Integer	Current region number being processed.
ISHIFT	Integer	Computed number of bits to shift word.
ITALT	Integer	Target altitude in feet above sea level.
ITERR	Integer	Switch for terrain data. 1=terrain data. o=no terrain data. Default is one. This variable is part of the parameter list.
ITREG	Integer	Region number of target

Label	Type	Description
ITX,ITY	Integer	Computed number of regions in X and Y directions
IX, IY	Integer	Computed column and row of lower corner region of radar range
IXCOR, IYCOR	Integer	Location of target within region
K1, K2	Integer	Pointers into the culture data buffer
LDATA	Integer Array	Array for parameter output record of integer data values
MOVE	Integer Array	Number of bits to shift word
MSIX	Integer Array	Index directories used by mass storage I/O
NBASER	Integer	Starting region number of map relative to the southwest corner
NR	Integer	Number of regions of radar coverage
NX	Integer	Maximum number of regions in X direction. This variable is part of the parameter list.
RANGE	Real	Radar radius of coverage in nautical miles. This variable is part of the parameter list.
RES	Real	Resolution unit size (in feet)
REGNM	Real	Regions per nautical mile
REST	Real	Resolution unit size (in meters X 1000). This variable is part of the parameter list.
TARGALT	Real	Target altitude in feet. This variable is part of the parameter list
TABLE	Real Array	Array for input terrain record packed four heights per word.
X, Y	Real	Radar location in meters from SW origin. These variables are part of the parameter list.

Label	Type	Description
WORD	Integer Array	Array for two word file output from RSS2.
XLON, YLAT	Real	X-Y target coordinates computed relative to the northwest corner.
YDIST	Real	Distance in Y-direction of the target in nautical miles from the northwest corner of the map.

2.2.7.7 Input. RSS7 requires three disk files and one card input file. The disk files required are the output terrain file (Tape 1) from RSS6, the output culture file (Tape 3) from RSS5, and the two word file (Tape 20) output from RSS2. The card input supplies certain NAMELIST quantities. The NAMELIST variables are as follows:

IFREQ, ITERR, ALTTDE, X, Y

See the program description for details of these variables.

2.2.7.8 Output. RSS7 outputs two disk files. The first file (Tape 6) is a parameter file. It contains two five-word records containing the descriptive information required in the construciton of the radar scene. Record one contains the following REAL variable data:

Word 1	RANGE
Word 2	ALTTDE
Word 3	X
Word 4	Y
Word 5	TARGALT

Record two contains the following INTEGER variable data:

Word 1	NX
Word 2	REST
Word 3	IFREQ
Word 4	IK
Word 5	ITERR

See the program description for details of these variables.

The second file (Tape 12) is a random-access file containing merged planimetry-terrain data records. These records are labeled by region number and have the following format:

Word 1	Region number
Word 2	Descriptive information for the N strips contained in the region. Each word contains the information for one strip, as output by RSS5.
Word N+1- Word N+IK^2	Packed elevation data as output by RSS7. A total of (IK^2/4+1) words are required to store the (IK X IK) elevation values for each region when packed four per word. The value of IK is dependent on the variable region size as defined by RSS2. The ordering of this elevation data and its relation to the corresponding grid locations within the region is the same as that output from RSS6. Figure II.7.3 shows a sample printout from RSS7.

2.2.7.9 Externals. System routines called by RSS7 are OPENMS, READMS, WRITMS, CLOSMS, and UNIT which are mass storage I/O routines. RSS7 calls no other subroutines.

2.2.7.10 Error Conditions. RSS7 has three error messages. The first error message is output if the resolution values of the terrain data and planimetry data are different. Its format is:

ERROR-CULTURE RESOLUTION NOT EQUAL TO TERRAIN RESOLUTION

The second error message is output if an error is found during input/output. Its format is:

HIT EOF OR PARITY ON I/P TAPE

The third error message is output if the terrain file is wholly or partially empty. Its format is:

END OF TERRAIN INPUT FILE REACHED ON REGION _____.

```
RLMS7 DATA EXTRACT FOR THE FOLLOWING RADAR PARAMETERS:
 HEIGHT DEPRESSION ANGLE RADIUS OF COVERAGE          LOCATION(X,Y)
  (FT)        (DEGR)              (N.M.)                  (N.M.)
 32000.0       3.34               19.194            20.0956   21.8591
(NMBR OF REGIONS OF COVERAGE= 46      STARTING REGION NMBR= 995)

CULTURE DATA FOR THE FOLLOWING REGIONS HAS OUTPUT:
(IN FORM -REGION  NWDS-)
```

996	37	1003	36	1012	25	1013	39	1017	1	1018	20	1019	23	1025	32	1027	18	1028	19
1031	18	1932	16	1035	32	1036	35	1038	34	1327	32	1328	6	1331	2	1332	31	1333	22
1335	12	1336	1	1344	55	1348	20	1355	11	1356	21	1357	11	1358	21	1363	5	1364	21
1355	5	1366	32	1367	35	1369	35	1372	12	1659	40	1664	29	1665	15	1667	23	1668	9
1675	64	1686	32	1688	32	1690	8	1691	3	1696	38	1697	27	1698	44	1699	29	1700	44
1701	9	1702	25	1703	42	1988	36	1989	28	1990	30	1991	28	1994	22	1995	19	1995	25
1999	32	2006	64	2017	33	2018	11	2019	46	2020	45	2021	2	2022	16	2023	9	2024	14
2025	5	2026	11	2027	36	2028	4	2029	53	2030	56	2031	20	2032	69	2033	35	2034	25
2319	32	2321	26	2322	12	2325	26	2326	12	2330	14	2331	18	2337	36	2338	28	2347	32
2349	30	2350	32	2351	32	2358	35	2359	5	2360	21	2361	30	2362	103	2363	64	2364	25
2650	46	2651	28	2652	31	2653	28	2654	30	2655	9	2656	42	2657	3	2662	32	2663	32
2669	32	2675	12	2677	37	2578	22	2680	16	2681	16	2682	32	2689	32	2691	32	2693	52
2694	55	2695	28	2695	9	2981	32	2986	14	2987	47	2993	14	2994	18	2999	32	3000	32
3006	24	3007	8	3008	31	3011	32	3012	12	3013	20	3020	32	3022	36	3024	32	3025	43
3026	46	3027	29	3312	32	3318	41	3325	50	3326	12	3327	1	3330	32	3331	11	3332	21
3337	43	3338	20	3341	4	3342	28	3343	32	3351	32	3353	32	3355	32	3357	64	3358	42
3643	32	3649	56	3656	61	3657	3	3658	8	3659	11	3660	10	3661	36	3662	8	3663	25
3664	23	3665	12	3665	6	3667	31	3668	38	3672	32	3673	14	3674	18	3692	32	3683	31
3684	6	3685	2	3686	32	3688	12	3689	91	3974	32	3977	6	3978	32	3979	51	3983	41
3986	31	3987	1	3988	32	3992	32	3995	27	3996	18	3997	36	3998	19	4003	32	4004	32
4012	21	4013	12	4014	32	4016	2	4017	41	4018	9	4019	40	4020	32	4305	45	4305	27
4307	47	4308	25	4309	49	4317	32	4319	13	4320	19	4323	32	4325	9	4326	6	4327	46
4328	40	4333	31	4334	31	4335	2	4343	32	4345	32	4348	32	4349	23	4350	9	4351	32
4636	9	4637	34	4648	32	4651	29	4652	3	4653	2	4654	66	4655	14	4656	3	4658	41
4659	29	4660	20	4661	8	4664	48	4665	16	4673	20	4674	12	4675	26	4676	6	4679	32
4680	32	4682	32	4978	28	4979	4	4983	46	4985	64	4987	12	4988	38	4989	4	4990	32
4992	13	4993	12	4994	1	4995	24	5003	6	5004	26	5005	13	5006	19	5007	26	5009	12
5009	5	5010	63	5013	32	5309	32	5314	48	5315	29	5316	64	5318	18	5319	14	5321	32
5334	32	5336	36	5337	12	5338	5	5339	2	5340	24	5341	49	5344	32	5631	6	5640	32
5641	2	5642	16	5643	27	5644	14	5645	111	5646	62	5647	87	5648	31	5649	29	5652	34
5653	18	5664	29	5665	3	5666	31	5667	1	5669	14	5670	18	5671	35	5672	32	5675	32
5962	7	5963	27	5969	8	5970	41	5971	34	5972	25	5973	19	5976	59	5977	24	5978	80
5979	32	5980	10	5981	13	5982	11	5983	32	5984	11	5985	7	5986	19	5994	13	5995	19
5997	32	5999	20	6000	12	6002	45	6003	19	6006	32	6291	2	6292	6	6293	38	6300	32
6301	31	6302	22	6303	7	6308	65	6309	72	6310	7	6313	3	6314	44	6317	32	6323	3
6324	27	6325	35	6328	32	6329	16	6330	41	6332	4	6333	60	6337	12	6624	1	6625	31
6631	17	6632	15	6633	32	6639	32	6640	67	6645	46	6646	7	6647	19	6648	13	6654	32
6656	32	6659	47	6660	18	6661	32	6662	7	6663	25	6664	32	6956	32	6958	13	6959	4
6960	5	6961	14	6962	10	6963	39	6964	34	6970	52	6971	43	6975	14	6976	18	6977	32
6978	32	6984	13	6985	5	6987	32	6988	4	6989	26	6990	9	6991	16	6992	16	6993	32
6994	12	6995	3	7287	16	7288	16	7289	32	7291	19	7294	32	7295	32	7301	63	7302	32
7306	29	7307	3	7308	32	7309	32	7317	20	7318	3	7319	31	7322	32	7323	30	7324	2
7330	4	7619	32	7620	3	7625	32	7626	32	7631	18	7632	46	7633	32	7638	32	7639	62
7640	2	7645	10	7646	18	7648	16	7649	20	7652	29	7653	3	7654	32	7661	32	7950	12
7951	4	7952	15	7953	7	7956	32	7957	56	7962	32	7963	25	7964	39	7969	33	7970	38
7976	26	7977	5	7978	20	7979	41	7980	2	7983	32	7985	24	7991	11	7992	22	8284	8
8285	12	8286	31	8287	19	8288	88	8289	8	8292	22	8293	10	8295	59	8300	32	8301	14
8311	32	8314	28	8315	31	8321	17	8322	42	8323	14	8617	32	8619	64	8620	32	8623	32
8626	64	8631	32	8642	32	8645	32	8651	22	8652	44	8654	22	8948	32	8950	38	8951	32
8953	20	8954	12	8955	2	8956	7	8957	68	8958	19	8959	6	8960	12	8961	10	8962	22

FIGURE II.8 - SAMPLE PRINTOUT FROM RSS7

```
(GROSS) GRID PARAMETERS; RADAR AT COL=  781 ROW=  850
 BEAM DISTANCE (RANGE)= 746 GRID ELEMENTS
 INITIAL SWEEP ANGLE OFFSET=    0.0 DEGREES
 RLMS8 SUCCESSFUL END
```

FIGURE II. 9 - SAMPLE PRINTOUT FROM RSS8

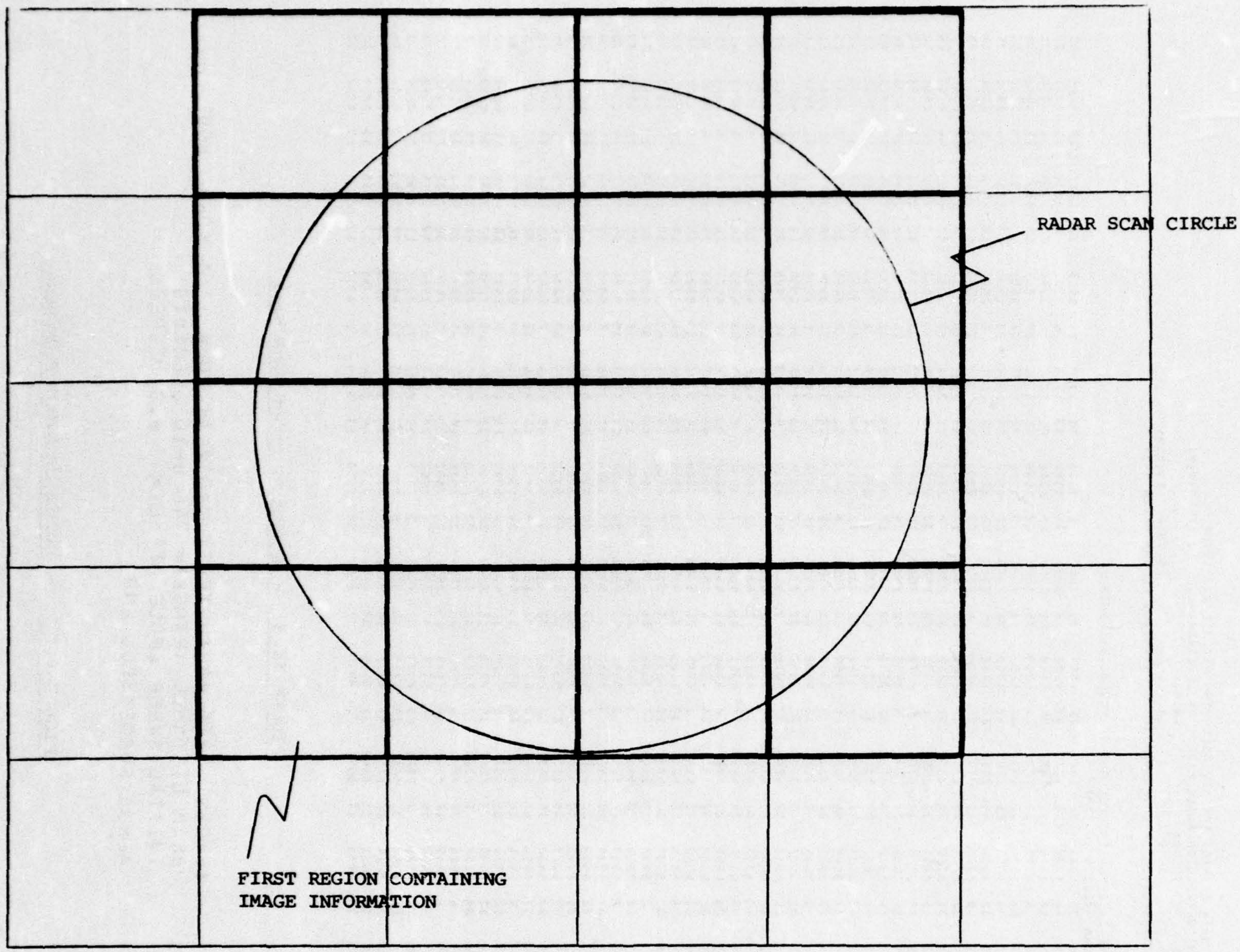


FIGURE II.10 - EXAMPLE OF REGION SUBSET EXTRACTED BY RSS7

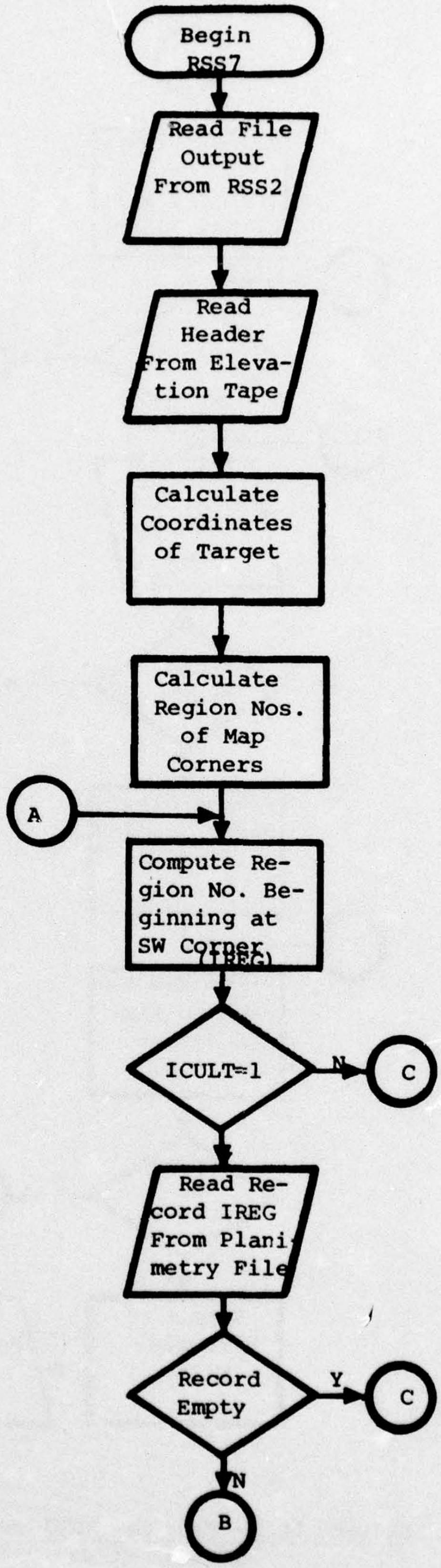


FIGURE II.11 - PROGRAM RSS7 FLOWCHART
(Page 1 of 2)

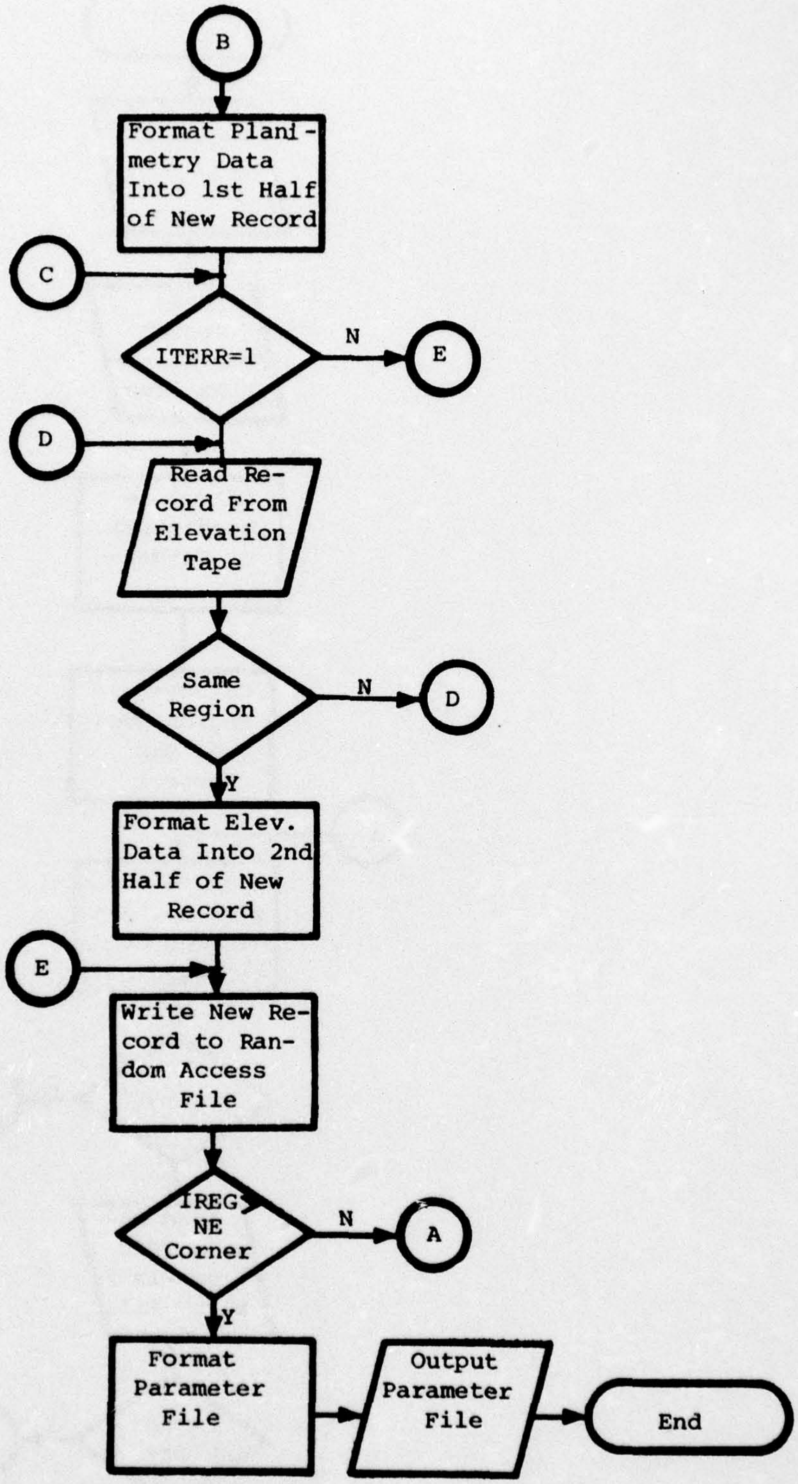


FIGURE II.11 - PROGRAM RSS7 FLOWCHART
(Page 2 of 2)

2.2.7.11 Program Flowchart. Figure II. 11 shows the flow diagram of RSS7.

2.2.7.12 Program Listing. The complete listing of RSS7 is attached.

2.2.8

2.2.8.1 Program Name. RSS8

2.2.8.2 Storage. Approximately 43K words of core

2.2.8.3 Run Time. Approximately 1800 cp seconds and 45 minutes wall clock time.

2.2.8.4 Cost. Approximately 375 dollars.

2.2.8.5 Program Function. RSS8 creates the radar scan lines from which the final radar scene is constructed.

2.2.8.6 Program Description. In polar coordinates, any point on the map may be described by the following expression:

$$X_p = X_t + r_p \text{ COS}\theta$$

$$Y_p = Y_t + r_p \text{ SIN}\theta$$

where (X_t, Y_t) is the location of the target, r_p the radial distance from the target, and θ the polar angle. The quantity IFREQ defines the values of θ to be used in the generation of the scan lines. These values are:

$$\theta_n = n/\text{IFREQ}, \quad 0 \leq n \leq 360*\text{IFREQ}$$

and result in the generation of a circle consisting of 360*IFREQ scen lines separated by an angular increment of 1/IFREQ degrees. The radar range R is expressed in units represented by variable RESK, the grid spacing for the map data. The variable r_p therefore has the range $0 < r_p < R$. At present the spacing for terrain and planimetry data must be equal. However an enhancement which allows for different resolution values has been coded into the program.

With the parameters thusly defined, the generation of the sweep lines proceeds as follows. We assume that the initial value of $\theta_n = 0$, i.e., n=0 and that we begin each scan line $r_p=1$.

A. Calculate $COS\theta_n$ and $SIN\theta_n$

B. Calculate X_p and Y_p

C. Calculate the region in which (X_p, Y_p) lies

D. *If the record for this region is already in core, go to (G)*

E. Read the record for this region into core

F. Copy the planimetry strip information for this region into IK*IK core array which serves as an image of the region. Radar return intensity levels are assigned in correspondence with the feature codes by using an internal translation table. The radar return intensity for resolution elements not covered by planimetry data is set equal to a predetermined background intensity. The result is that array (I,J) contains the radar return intensity (from planimetry only) for the resolution element at X=I, Y=J.

G. Calculate which resolution element contains (X_p, Y_p). Denote this by ARRAY (I_p, J_p).

H. Copy ARRAY (I_p, J_p) and the corresponding terrain value from the input region record into a linear array which stores the information for the scan line.

I. Increment r by 1.

J. If $r \leq R$ go to (B)

K. Increment θ by 1/IFREQ

L. If $\theta > 360^{\circ}$ we are finished. If $\theta \leq 360^{\circ}$ go to (A)

The following is a description of the important variables in RLMS8.

Label	Type	Description
ALPHA	REAL	Starting angle of radar swap-scan
ARRAY	INTEGER ARRAY	Elevation values of each point in the terrain region

Label	Type	Description
ALTTDE	REAL	Altitude of radar in feet above ground level.
CONV	REAL	Conversion factor. Computes radians per degree
DCONV, ECONV	REAL	Conversion factor. Computes grid points per nautical mile
CTH	REAL	COS of sweep angle
DATA	INTEGER ARRAY	Feature intensity level conversion table
DLINK	REAL ARRAY	Array for parameter input record of REAL data values
HALFX, HALFY	REAL	Rounding factor. Value is .5
IA, IAA, IB, IBB	INTEGER	Row and column of current gross element number under beam scan
IANGLE	INTEGER	Scan rotation angle counter
IANX, IANY	INTEGER	Polar position of point within region
IARRAY	INTEGER ARRAY	Intensity codes for each point in the culture region
IBACK	INTEGER	Intensity code for background of picture
ID	INTEGER	Number of grid elements in a line sweep. Maximum value is 2000.
IFREQ	INTEGER	The number of radial scan lines to be constructed per degree. This variable is part of the parameter list.
IHT	INTEGER	Elevation value for point along radial scan line.
II	INTEGER ARRAY	Array of assigned intensity codes for each feature
IFL	INTEGER	Bit pointer into II array

Label	Type	Description
IIP	INTEGER	Bit pointer to elevation value
IIX, IIY	INTEGER	Word pointers to elevation value
IK	INTEGER	Region size for terrain data
IKC	INTEGER	Region size for culture data
IN	INTEGER	Number of words for region row data
ILIM	INTEGER	Number of words in current region record
IMIN, IMAX	INTEGER	Beginning and ending scan angle (+1)
INR	INTEGER	Equivalent to REG used as an index for READMS.
IN2	INTEGER	Intensity code for point along radial scan line
IN2W	INTEGER	Input buffer for two word file output by RSS2.
IXL, IYL	INTEGER	Row and column element values
ITERR	INTEGER	Switch for terrain data. 1= terrain data. 0= no terrain data. This variable is part of the parameter list.
JLIM	INTEGER	Number of data values region row.
KX, KY	INTEGER	Row and column of current element under beam scan within a given culture region
KKX, KKY	INTEGER	Row and column of current element under beam scan within a given terrain region
MSIX	INTEGER ARRAY	Index directories used by mass storage I/O.
NLINK	INTEGER ARRAY	Array for parameter input record of INTEGER data values
NRX	INTEGER	Maximum number of allowable regions along X-AXIS.

Label	Type	Description
NREG	INTEGER	Region number for which data is currently in core.
P	REAL	Current angle of sweep line in degrees.
RANGE	REAL	Radar radius of coverage in nautical miles. This variable is part of parameter list.
REG	INTEGER	Region number of current sweep line element.
RESK	REAL	Resolution unit size in feet.
SCNLNE	INTEGER ARRAY	Scan line storage for one row of culture intensities and corresponding elevations
STH	REAL	SIN of sweep angle
THETA	REAL	Current angle of sweep line in radians
WORD	INTEGER ARRAY	Array for two word file output from RLMS2
XI, YI	INTEGER	Radar location in terms of gross data grid
X, Y	INTEGER	Target location

2.2.8.7 Input. RSS8 requires three input disk files. The files required are the two word output file (Tape 20) from RSS2, the output parameter file (Tape 6) from RSS7 and the random-access map data file (Tape 12) from RSS7,

2.2.8.8 Output. RSS8 outputs one disk file (Tape 4). This file consists of 360*IFREQ records of 4000 words each. The file is sequential in nature with the records being ordered by their corresponding angle variable θ, i.e. the record for scan line at $\theta=0^{\circ}$ is the first record on the file while that for $\theta=360^{\circ}$ is the last record.

The first 2000 words contain the intensity-of-return values for each r, $0<r\leq R$. Clearly, if $R<2000$ the remaining words are blank. Words numbered 2000+I, $1\leq I\leq 2000$, contain the corresponding terrain elevation

values. For example, if IFREQ=2, then the fourth record has the following significance.

A. It corresponds to $\theta=4/IFREQ=2^{\circ}$

B. The I^{th} word and the $I+2000^{th}$ word respectively contain the strength of return and the terrain elevation for the point:

$$X_p = X_t + I*\cos 2^{\circ}$$

$$Y_p = Y_t + I*\sin 2^{\circ}$$

The printout from RSS8 is shown in Figure II.8.2 and is self-explanatory.

2.2.8.9 Externals. System routines called by RSS8 are three mass storage I/O function routines OPENMS, READMS, and UNIT; and three trigonometric function routines ATAN2, COS, and SIN. RSS8 calls no other subroutines.

2.2.8.10 Error Conditions. RSS8 has three error conditions. Error condition number one occurs when there are more than 500 culture strips for any given region. The format of the error message for this condition is as follows:

VALUE COUNT=________, IN ERROR FOR REGION_____

....OCCURRED DURING SWEEP ANGLE OF _____ DEGREES

Error condition number two occurs when the index for the region data extends beyond the region bounds. The error messages for this condition are as follows:

COL ELEMENT VALUE=_____, IN ERROR FOR REGION_____

....OCCURRED DURING SWEEP ANGLE OF ____ DEGREES

OR

ROW ELEMENT VALUE=____, IN ERROR FOR REGION______

....OCCURRED DURING SWEEP ANGLE OF _____DEGREES

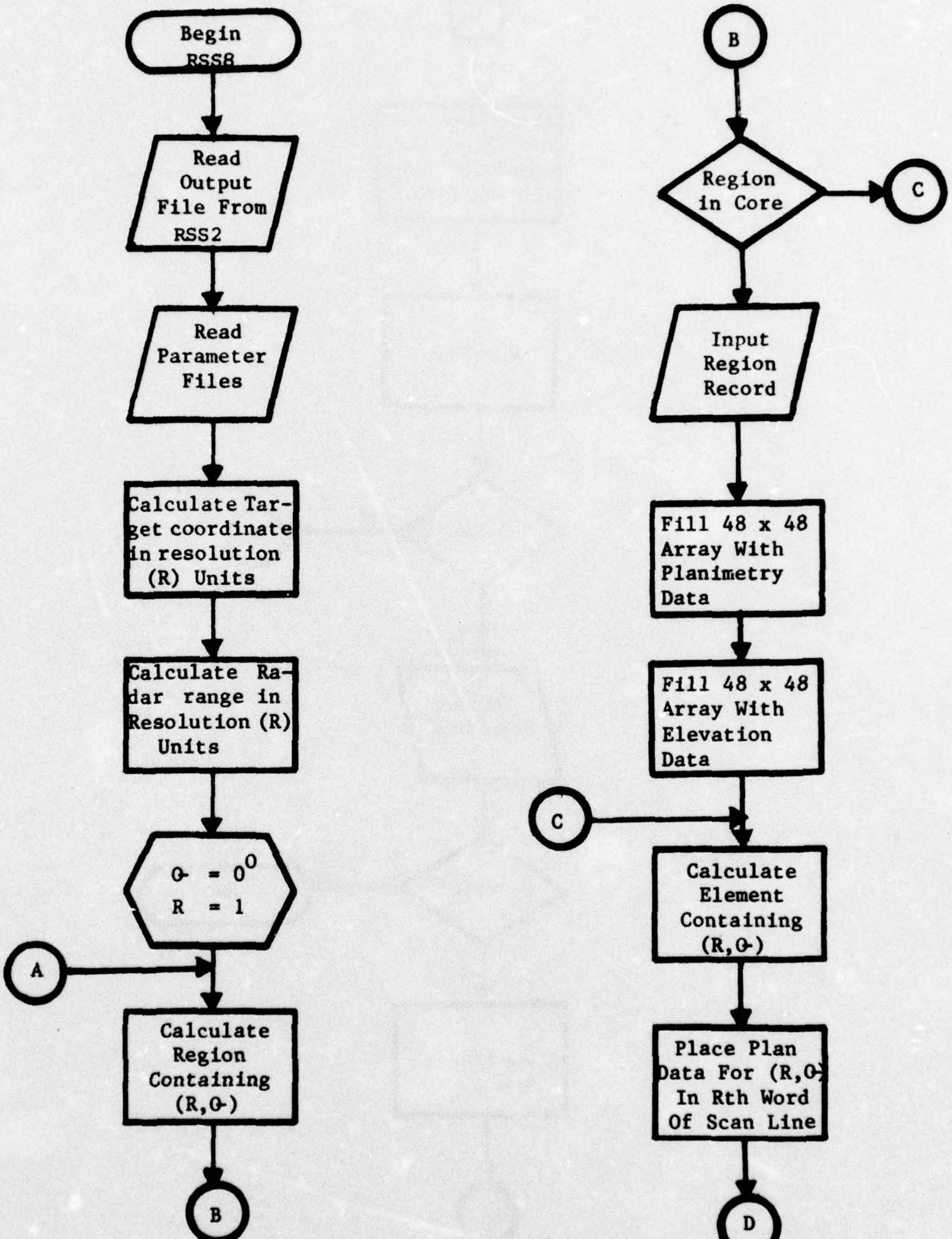


FIGURE II.12- PROGRAM RSS8 FLOWCHART
(Page 1 of 2)

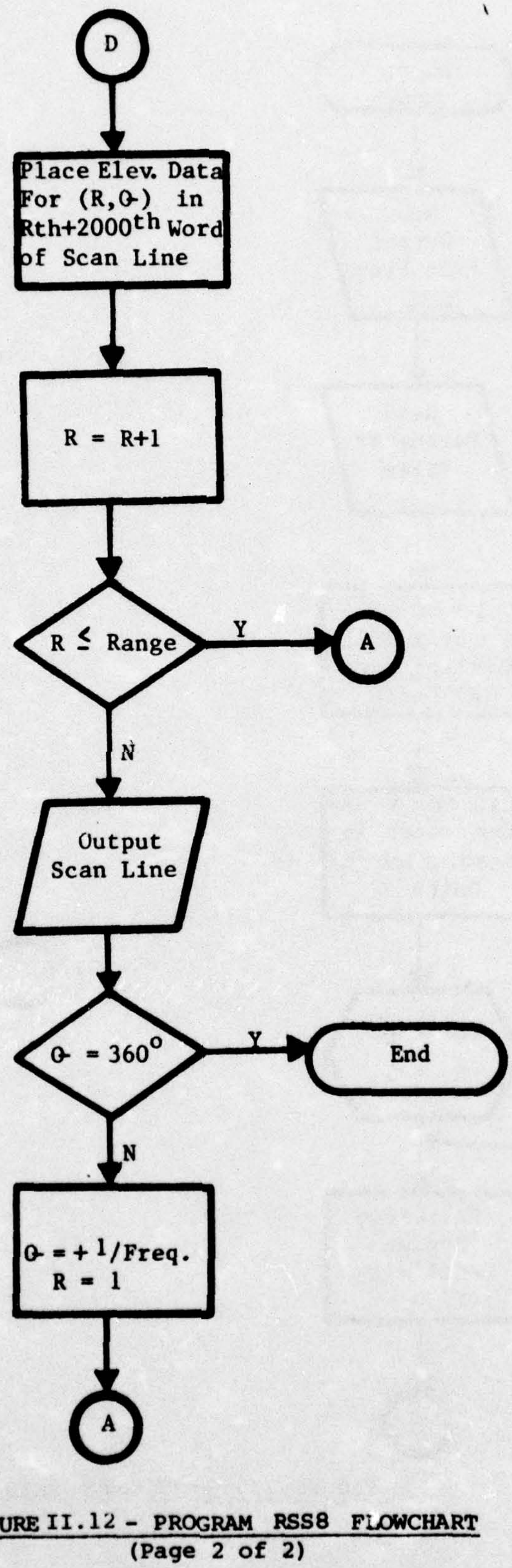


FIGURE II.12 - PROGRAM RSS8 FLOWCHART
(Page 2 of 2)

The third error condition occurs if there is an error found during input/output. The format of its error messages is as follows:

EOF OR PARITY ERROR ON UNIT 6, RSS8

....OCCURRED DURING SWEEP ANGLE OF ____ DEGREES

OR

EOF OR PARITY ERROR ON O/P FILE 4

....OCCURRED DURING SWEEP ANGLE OF ____ DEGREES

OR

INPUT ERROR ON UNIT 20

2.2.8.11 Program Flowchart. Figure II. 12 shows the flow diagram of RSS8.

2.2.8.12 Program Listing. The complete listing of RSS8 is attached.

2.2.9

2.2.9.1 Program Name. RSS9

2.2.9.2 Storage. Approximately 10K words of core

2.2.9.3 Run Time. Approximately 1000 cp seconds and 40 minutes wall clock time

2.2.9.4 Cost. Approximately 160 dollars

2.2.9.5 Program Function. RSS9 applies the radar effects to the radial scan line data from RSS8.

2.2.9.6 Program Description. The three radar effects considered by RSS9 are:

(1) Lambert's Law effect which determines the percentage of the incident radar signal reflected back to the source.

(2) Shadow effect which takes into account the fact that certain areas on the ground may be blocked from view by tall objects

(i.e. mountain peaks) lying in the line-of-sight from the radar location to the area in question.

(3) Altitude layover effect which takes into account that the radar perceives the location of high altitude terrain to be closer to the radar than it actually is, and below sea level terrain to be farther from the radar than it actually is.

RLMS9 contains a subroutine used to scale the final image so that scenes generated from different altitudes will all be the same size when displayed on the DICOMED plotter. RSS9 also contains a subroutine which converts the radial formatted points back to cartesian coordinates for eventual display. The following is a description of the important variables in RSS9.

Label	Type	Description
ALT	INTEGER	Altitude of the radar above sea level (in feet)
ATARG	REAL	Altitude of the radar above the target (in feet)
AVINS	REAL ARRAY	Array which contains the average of intensity for each point along the scan line, i.e., each value in the array equals $\frac{\sum_{r=1}^{n} I_r}{720}$ where n is the number of points along the scan line, and I is the intensity value for the point r under consideration; and 360°*IFREQ=720.
CONV	REAL	Conversion factor. Computes radians per degree.
DATA	INTEGER ARRAY	Array for input data record of combined culture and terrain values by sweep line
ELEV	INTEGER ARRAY	The 2001^{th}-4000^{th} word of the input data which contains the terrain values.
ENTS	REAL	The intrinsic intensity value for a point with Lambert's Law included as a factor.

Label	Type	Description
ICODE	INTEGER ARRAY	Array of counters. Counts the number of each color code generated before Lambert's Law is considered.
ICOLOR	INTEGER ARRAY	Array of counters. Counts the number of each color code generated <u>before</u> altitude layover is considered.
ICOLO2	INTEGER ARRAY	Array of counters. Counts the number of each color code generated <u>after</u> altitude layover is considered.
IDIST	INTEGER	Number of points along the radial scan line.
IFREQ	INTEGER	The number of radial scan lines to be constructed per degree.
IMIN, IMAX	INTEGER	Beginning and ending scan angle (+1)
INS	INTEGER ARRAY	The 1^{st}-2000^{th} word of the input data which contains the culture values.
IN2W	INTEGER	Input buffer for two word file output by RSS2.
ISHADE	INTEGER	Counter for the number of points computed to be a shadow point by subroutine LAMBERT
ITEMP	INTEGER	Save variable for IDIST
KOUNT	INTEGER	Counter for the number of radial sweep points processed.
MM	INTEGER	Sweep angle (+1)
RESK	REAL	Resolution unit size in feet for the culture data
REST	REAL	Resolution unit size in feet for the terrain data
RET	REAL ARRAY	Array containing the predetermined intensity strengths for intensity codes 0-63
SCALE	REAL	Scaling factor. Scales the radar picture up or down. SCALE=IDIST/1000

Label	Type	Description
SR	REAL ARRAY	Array for the strengths of return as computed by subroutine LAMBERT
THETA	REAL	Current angle of sweep line in radians
WORD	INTEGER ARRAY	Array for two word file output by RSS2.

2.2.9.7 Input. RSS9 requires three input disk files and one input card file. The files required are the two word output file (Tape 20) from RSS2, the output parameter file (Tape 6) from RSS7, and the radial scan line data file (Tape 4) from RSS8. The card file is a one word parameter file which contains the altitude of the radar above the target.

2.2.9.8 Output. RSS9 outputs four disk files (Tape 3, Tape 10, Tape 12, and Tape 14). These files contain the data for unsorted raster converted points output by subroutine REFRMT. Each output file contains 180000 two word records. These 180000 records represent data for one quarter of the picture, i.e., 90*IFREQ*1000=180000. Each record represents data for a single point on the image. Its format is as follows:

Word 1	MOD(X, Y) coordinate in units represented by RESK
Word 2	Intensity values from 0-63. Here intensity zero corresponds to a maximum radar return (white) while a 63 corresponds to zero return. This system is the same as used on the DICOMED plotter.

The printout generated by RSS9 is shown in Figure II. 14 The input statistics indicate the distribution of color codes as output by RSS8. In this case the color code 2 was used for the background. The output statistics indicated the distributions of color codes output after execution of subroutine LAMBERT and then subroutine ALTLAV.

```
RENS9 SUCCESSFUL END, DISPLAY FILE COMPLETE
INPUT STATISTICS
   133075 PIXELS WITH COLOR CODE  2
      649 PIXELS WITH COLOR CODE 10
      162 PIXELS WITH COLOR CODE 12
      394 PIXELS WITH COLOR CODE 15
OUTPUT STATISTICS
        5 PIXELS WITH COLOR CODE  8
       16 PIXELS WITH COLOR CODE  9
       29 PIXELS WITH COLOR CODE 10
       28 PIXELS WITH COLOR CODE 11
       60 PIXELS WITH COLOR CODE 12
       32 PIXELS WITH COLOR CODE 13
       49 PIXELS WITH COLOR CODE 14
       62 PIXELS WITH COLOR CODE 15
       73 PIXELS WITH COLOR CODE 16
       80 PIXELS WITH COLOR CODE 17
       85 PIXELS WITH COLOR CODE 18
       96 PIXELS WITH COLOR CODE 19
       94 PIXELS WITH COLOR CODE 20
      149 PIXELS WITH COLOR CODE 21
      165 PIXELS WITH COLOR CODE 22
      186 PIXELS WITH COLOR CODE 23
      239 PIXELS WITH COLOR CODE 24
      279 PIXELS WITH COLOR CODE 25
      316 PIXELS WITH COLOR CODE 26
      389 PIXELS WITH COLOR CODE 27
      502 PIXELS WITH COLOR CODE 28
      558 PIXELS WITH COLOR CODE 29
      687 PIXELS WITH COLOR CODE 30
      973 PIXELS WITH COLOR CODE 31
     1407 PIXELS WITH COLOR CODE 32
     2326 PIXELS WITH COLOR CODE 33
     3695 PIXELS WITH COLOR CODE 34
     4849 PIXELS WITH COLOR CODE 35
     5932 PIXELS WITH COLOR CODE 36
     9322 PIXELS WITH COLOR CODE 37
    13391 PIXELS WITH COLOR CODE 38
    21559 PIXELS WITH COLOR CODE 39
    30185 PIXELS WITH COLOR CODE 40
    15275 PIXELS WITH COLOR CODE 41
     7029 PIXELS WITH COLOR CODE 42
     3836 PIXELS WITH COLOR CODE 43
     2418 PIXELS WITH COLOR CODE 44
     1632 PIXELS WITH COLOR CODE 45
     1029 PIXELS WITH COLOR CODE 46
      790 PIXELS WITH COLOR CODE 47
      594 PIXELS WITH COLOR CODE 48
      437 PIXELS WITH COLOR CODE 49
      359 PIXELS WITH COLOR CODE 50
      331 PIXELS WITH COLOR CODE 51
      308 PIXELS WITH COLOR CODE 52
      569 PIXELS WITH COLOR CODE 53
      383 PIXELS WITH COLOR CODE 54
      207 PIXELS WITH COLOR CODE 55
      235 PIXELS WITH COLOR CODE 56
      161 PIXELS WITH COLOR CODE 57
      145 PIXELS WITH COLOR CODE 58
      102 PIXELS WITH COLOR CODE 59
      105 PIXELS WITH COLOR CODE 60
      106 PIXELS WITH COLOR CODE 61
      441 PIXELS WITH COLOR CODE 62
      970 PIXELS WITH COLOR CODE 63
      924 PIXELS WERE ASSIGNED TO THE SHADOW
```

FIGURE II.14 - SAMPLE PRINTOUT FROM RSS9

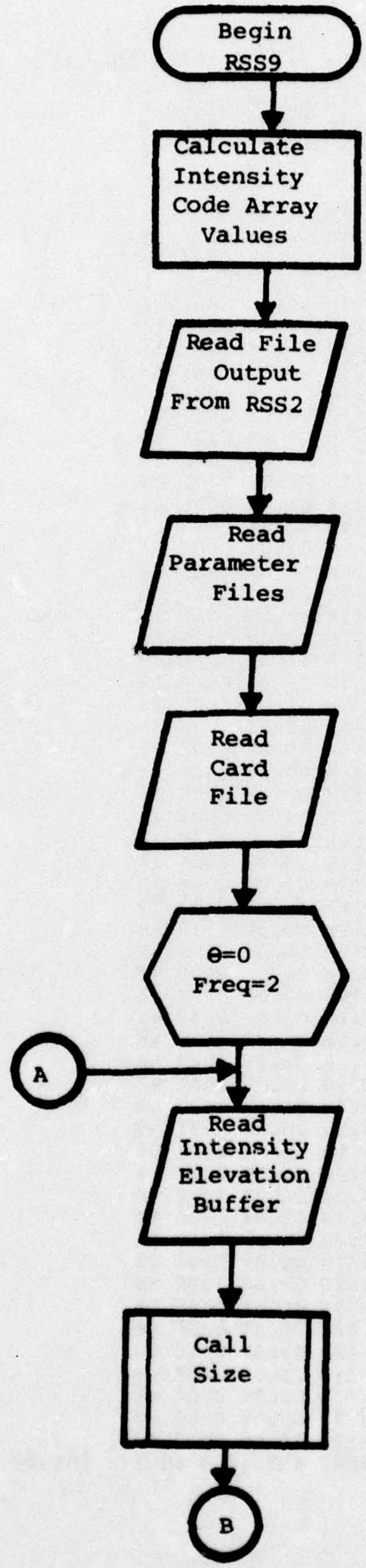


FIGURE II.15- PROGRAM RSS9 FLOWCHART
(Page 1 of 2)

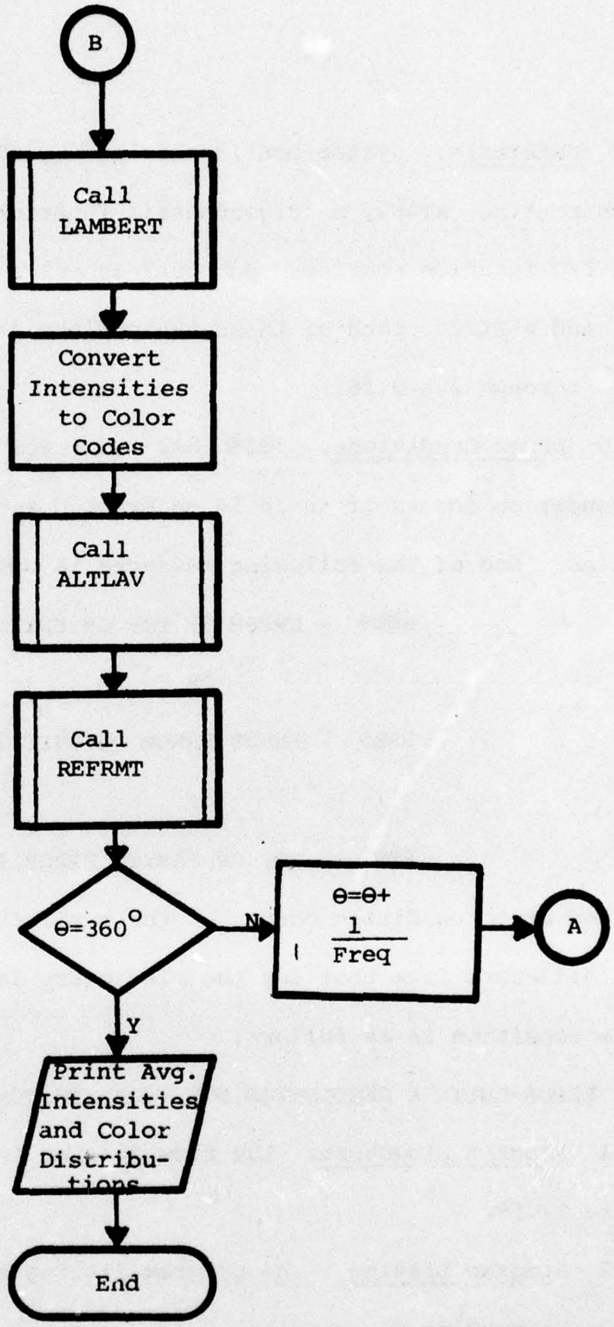


FIGURE II.15 PROGRAM RSS9 FLOWCHART
(Page 2 of 2)

2.2.9.9 Externals. System routines called by RSS9 are ALOG10, a logarithmic function routine; ATAN2, a trigonometric function routine; and UNIT, a mass storage I/O function routine. Subroutines called by RSS9 are LAMBERT, SIZE, REFRMT, and ALTLAV. Each of these subroutines are discussed under sections 2.2.9.13 through 2.2.9.16.

2.2.9.10 Error Conditions. RSS9 has two error conditions. The first error condition occurs if there is an error during input or output to the disk files. One of the following messages is output for this error condition:

RSS9 - ERROR OR EOF ON UNIT 6

OR

RSS9 - INPUT ERROR ON UNIT 20

or

RSS9 - EOF OR PARITY ERROR IN I/P FILE 4

The second error condition occurs if the resolution value for the terrain data is different from that for the planimetry data. The message output for this condition is as follows:

ERROR-CULTURE RESOLUTION NOT EQUAL TO TERRAIN RESOLUTION

2.2.9.11 Program Flowchart. The flow diagram for RSS9 is illustrated in Figure II.15.

2.2.9.12 Program Listing. The program listing for RSS9 is attached.

2.2.9.13 Subroutine Description

2.2.9.13.1 Subroutine Name LAMBERT

2.2.9.13.2 Summary. LAMBERT computes the Lambert's Law and shadow radar effects.

2.2.9.13.3 <u>Description of Processing</u>. The calling sequence for LAMBERT is LAMBERT (ELEV, ALT, SR, REST). For a description of the calling arguments see the list of important variables for this subroutine. The Lambert's Law effect is illustrated in Figure II.16.

The mathematical statement of this law is that the return from any given point on the ground is reduced by a factor COS θ_L where θ_L is the angle between the incident radar beam and the normal to the terrain at the point of interest. This is to say, if I is the intrinsic intensity of the background or of the planimetry feature located at a given point, then the actual radar return intensity is given by:

$$I_r = I*COS\theta_L$$

The implementation of this effect is quite straightforward. For each value of r, the slope of the line through the points r-1 and r+1 is calculated from the corresponding terrain elevation values thus yielding the angle θ_t. The vector perpendicular to this line defines the normal to the surface at r. The declination angle of the radar beam θ_D can be calculated from r, the terrain elevation at r, and the radar altitude. Using these angles, the angle θ_L can be determined from elementary trigonometry.

The shadowing effect is illustrated in Figure II. 17. The nature of the effect is simply that points on the ground may be in the shadow terrain peaks lying between the radar and the ground point of interest. Such points will be invisible to the radar and must therefore be assigned a zero intensity-of-return.

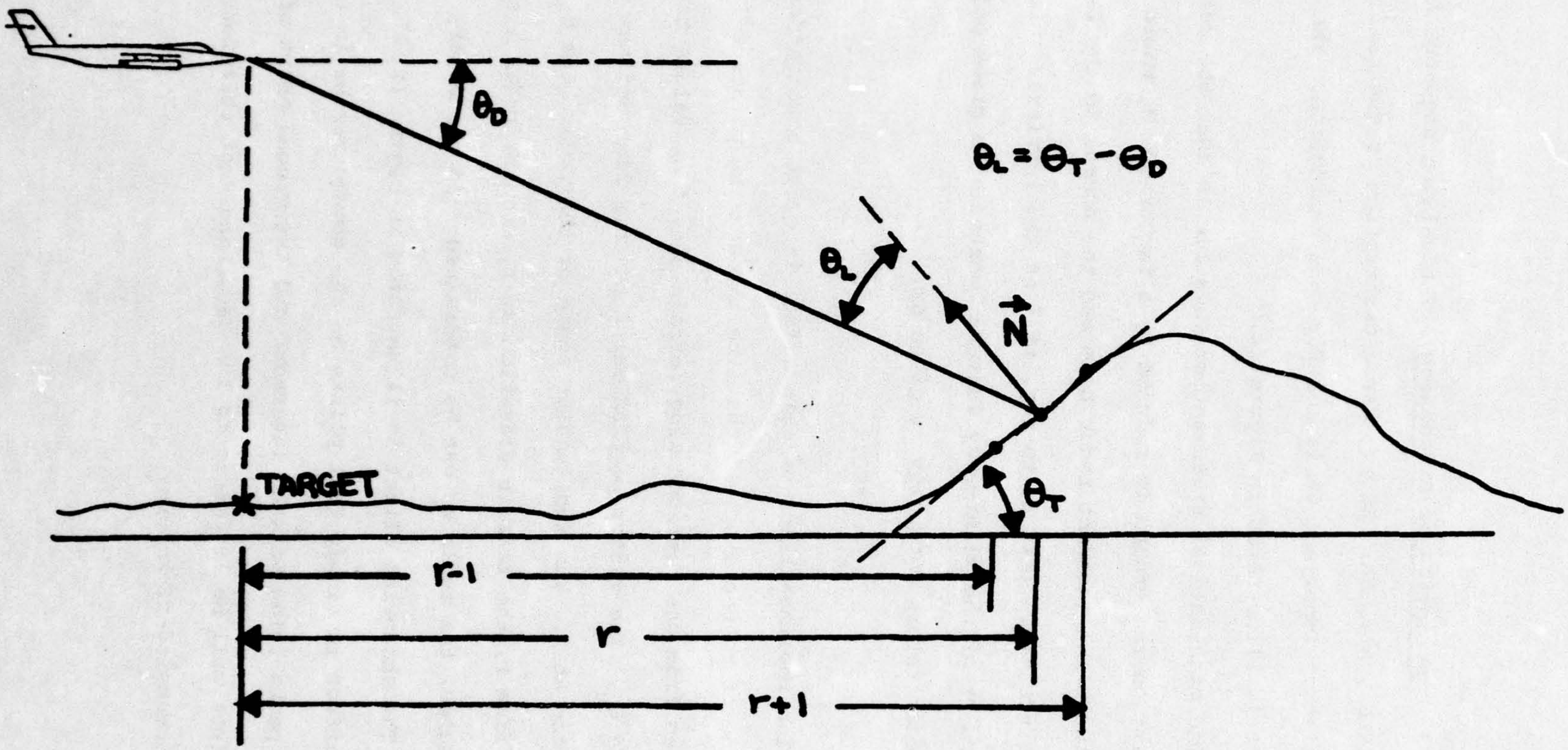


FIGURE II.16 - DEFINITIONS OF LAMBERT'S LAW QUANITIES

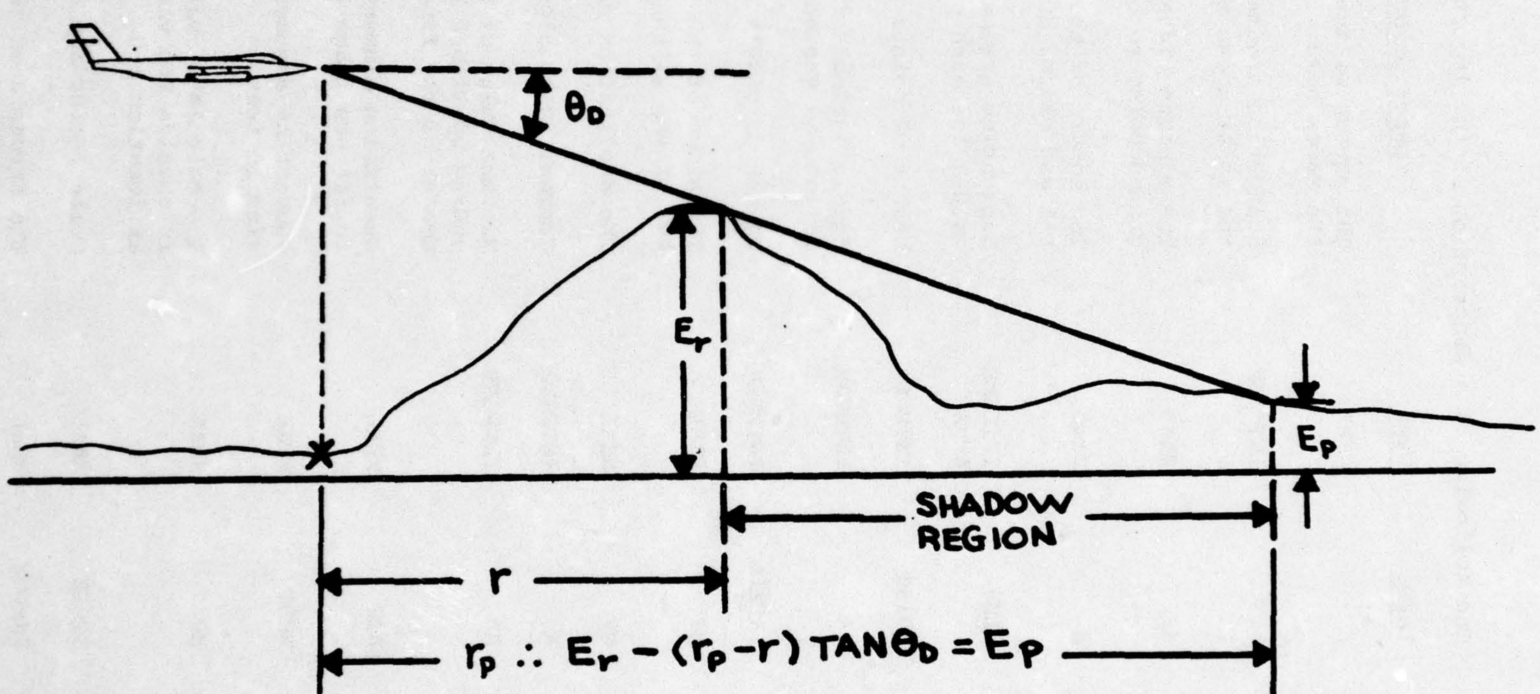


FIGURE II.17 - ILLUSTRATION OF SHADOW EFFECT

The following is a description of the important variables in LAMBERT.

Label	Type	Description
A	REAL	The square of the declination angle of the radar beam.
ALT	INTEGER	Subroutine argument. The altitude of the radar above sea level.
AR	REAL	The minimum elevation required for a ground point to be no longer in shadow.
B	REAL	The square of the slope of the line through r-1 and r+1 of any given point r.
ELEV	INTEGER ARRAY	Subroutine argument. Array of elevation values for each point along a radial line.
IDIST	INTEGER	Number of points per radial scan line
IR	INTEGER	Equals variable 'R' with a cutoff value defined by variable 'RD'.
ISHADE	INTEGER	Number of pixels assigned to shadow
MT	REAL	The slope of the line through r-1 and r+1 of any given point r.
MV	REAL	The declination angle of the radar beam
R	INTEGER	Current point along a radial scan line
RD	INTEGER	Minimum value of R in order to avoid ending up with a bright area in the center of the final image
RES	REAL	Resolution element size in feet scaled to fit new image size
REST	REAL	Subroutine argument. Resolution element size in feet.
RP	REAL	The calculated value of 'R' if point R is outside the view range of the radar at location r=0.
SCALE	REAL	Scale factor for adjusting the image size
SHADOW	REAL	The strength of return for a shadow point

Label	Type	Description
SLOPE	REAL	Same as 'MV'
SR	REAL ARRAY	Subroutine argument. Strengths of return for each radial point.
SRT	REAL	Computed strength of return for a point before being multiplied by the factor RD/500 to avoid getting a bright area in the center of the final image.
X	REAL	The factor RD/500 used to avoid getting a bright area in the center of the final image

2.2.9.13.4 Error Conditions. None

2.2.9.13.5 Subroutine Flowchart. The flow diagram for LAMBERT is illustrated in Figure II.18.

2.2.9.13.6 Subroutine Listing. The listing for LAMBERT is included with that of RLMS9.

2.2.9.14

2.2.9.14.1 Subroutine Name ALTLAV

2.2.9.14.2 Summary. ALTLAV computes the altitude layover radar effect.

2.2.9.14.3 Description of Processing. The calling sequence for ALTLAV is ALTLAV(ELEV, ALT, INS, ATARG, RET, SR, REST). For a description of the calling arguments see the list of important variables for this subroutine.

The altitude layover effect shifts the location of the terrain. The length and direction of the shifting of each processed point of the terrain depends on the height of that point. This effect is implemented by considering each point R along a radial line and computing a new altitude layover point R1. Given (1) the distance of the point (R) away from the target point along a radial line, (2) the height of the radar above sea level (A), and (3) the elevation of the point R above sea level (E); the

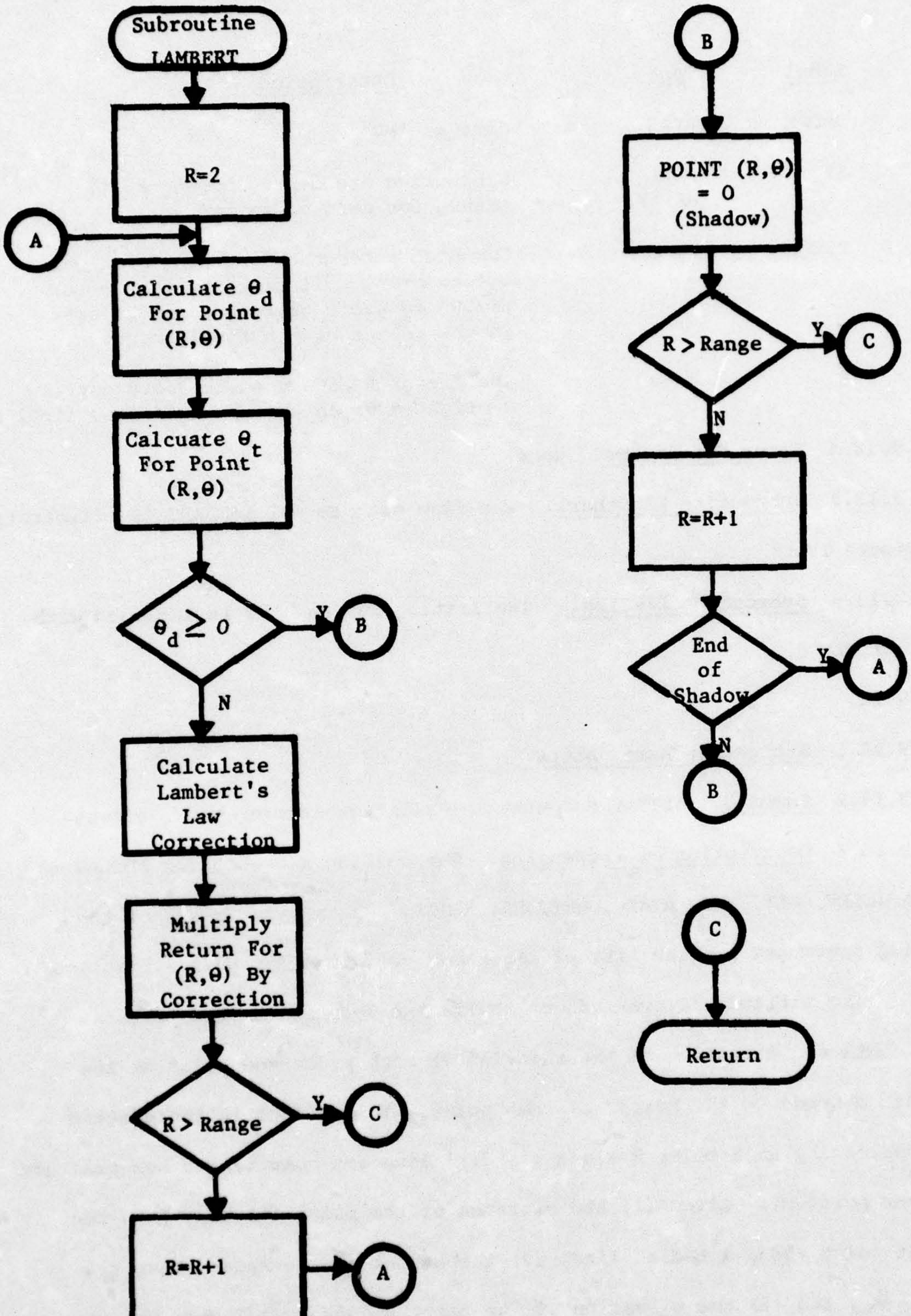


FIGURE II.18 - SUBROUTINE LAMBERT FLOWCHART

distance (D) from the radar to the elevated point along the radial can be computed as follows:

$$D = \sqrt{(A-E^2 + R^2}$$

Using a principle of elementary trigonometry, we know that a tangent line drawn the length of the elevation of point R to some point R1 at target level along the radial line yields a distance D1 equal in length to distance D. The point R1 can be computed similarly to the above equation. Having computed D=D1, and given the radar altitude ablve target level (ALT), the point R1 computed at target level is as follows:

$$R1 = \sqrt{ALT^2 + (A-E)^2 + R^2}$$

The following is a description of the important variables in ALTLAV:

Label	Type	Description
ALT	INTEGER	Subroutine argument. The altitude of the radar above sea level.
ATARG	REAL	Subroutine argument. The altitude of the radar above the target
DX	REAL	The difference in feet between the distance of a reference point from the target location and the calculated altitude layover distance from the target.
ELEV	INTEGER ARRAY	Subroutine argument. Array of elevation values for each point along a radial line
ENTS1	REAL	The density value for color code 0.
ENTS	REAL ARRAY	Array containing the density values for each point along radial line. When altitude layover requires two density values to be added, the sum is placed into this array.

Label	Type	Description
IDIST	INTEGER	Number of points per radial scan line
INS	INTEGER ARRAY	Subroutine argument. Intensity codes for each point along a radial scan line
ISHADE	INTEGER	Number of pixels assigned to shadow
RES	REAL	Resolution element size in feet scaled to fit new image size
REST	REAL	Subroutine argument. Resolution element size in feet
RET	REAL ARRAY	Subroutine argument. Density values of each of the 64 color codes used
RP	REAL	The calculated altitude layover distance in feet from target
RPM	REAL	Radial distance of reference point in feet from target
R1	INTEGER	Grid location along radial of the calculated altitude layover point
R2	INTEGER	Maximum value of r for points defined by (r, θ) that a radar positioned at r=0 can view
R3	INTEGER	Minimum value of r for points defined by (r, θ) that a radar positioned at r=0 can view
SR	REAL ARRAY	Subroutine argument. Strengths of return for each point calculated by subroutine LAMBERT
SCALE	REAL	Subroutine argument. Scale factor for adjusting the image size
XP	REAL	The distance in feet of a radial point from the vertex minus the value of DX
X	REAL	The calculated altitude layover distance in feet from the target location

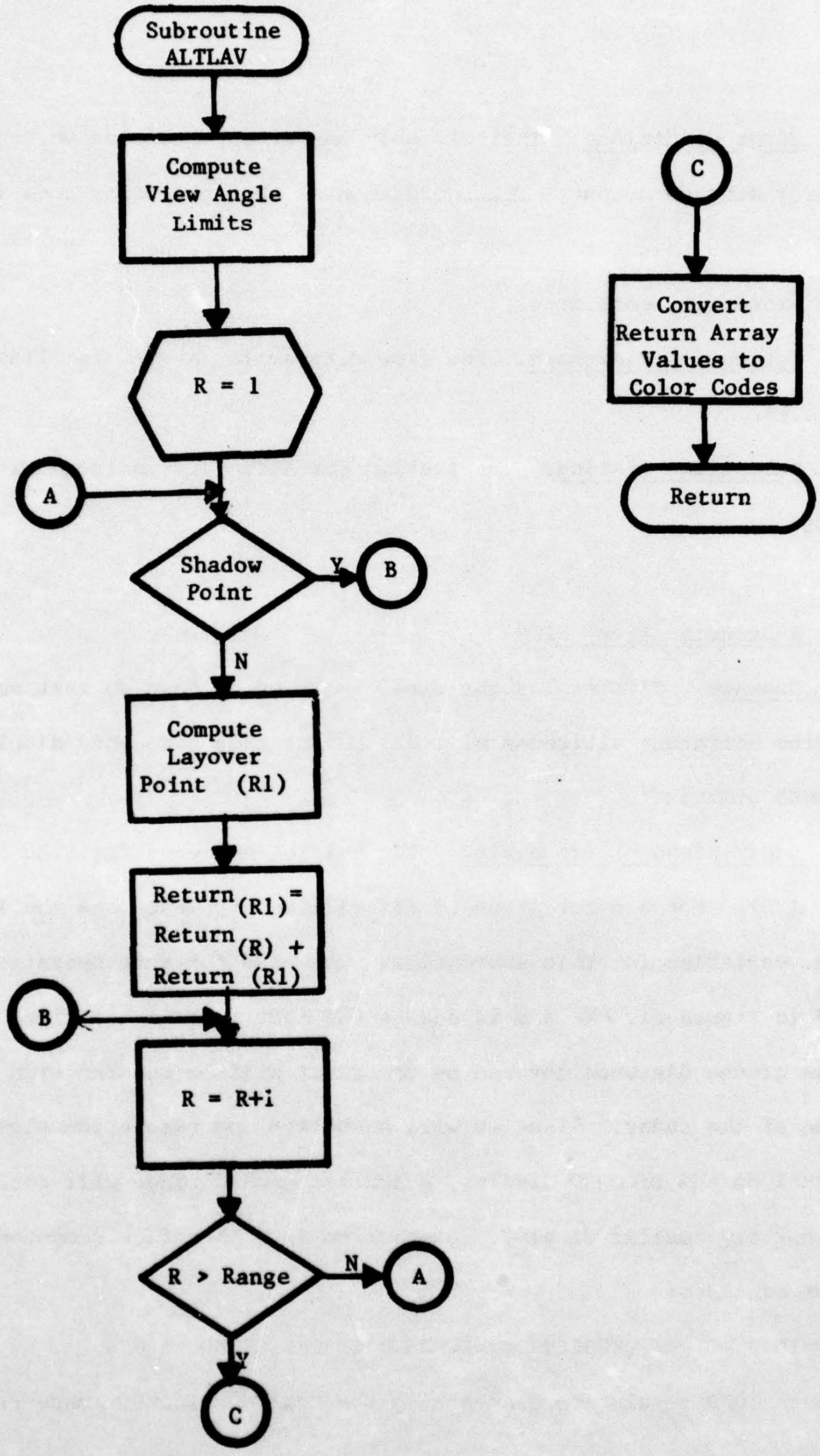


FIGURE II.19 - SUBROUTINE ALTLAV FLOWCHART

2.2.9.14.4 Error Conditions. There is only one error condition which will cause an error message output. That condition is when the color code is computed to be less than zero. When this occurs, the erroneous code is printed and processing continues.

2.2.9.14.5 Subroutine Flowchart. The flow diagram for ALTLAV is illustrated in Figure II.19.

2.2.9.14.6 Subroutine Listing. The listing for ALTLAV is included with that of RSS9.

2.2.9.15

2.2.9.15.1 Subroutine Name SIZE

2.2.9.15.2 Summary. SIZE scales the final image up or down so that scenes generated from different altitudes will all be the same size when displayed on the DICOMED plotter.

2.2.9.15.3 Description of Processing. The calling sequence for SIZE is SIZE (ELEV, INS). For a description of the calling arguments see the list of important variables for this subroutine. The need for this operation is illustrated in Figure II. 20, ~~and is due~~ to the fact that with a fixed radar aperture the ground distance covered by the radar will be the function of the altitude of the radar. Since we will associate one resolution element with one pixel on the DICOMED display, a smaller ground range will result in a progressively smaller display. Subroutine SIZE therefore computes a scale factor equal to:

F = 1000/NO. OF RESOLUTION ELEMENTS IN THE GROUND RANGE

We have chosen 1000 resolution elements as the desired DICOMED image radius,

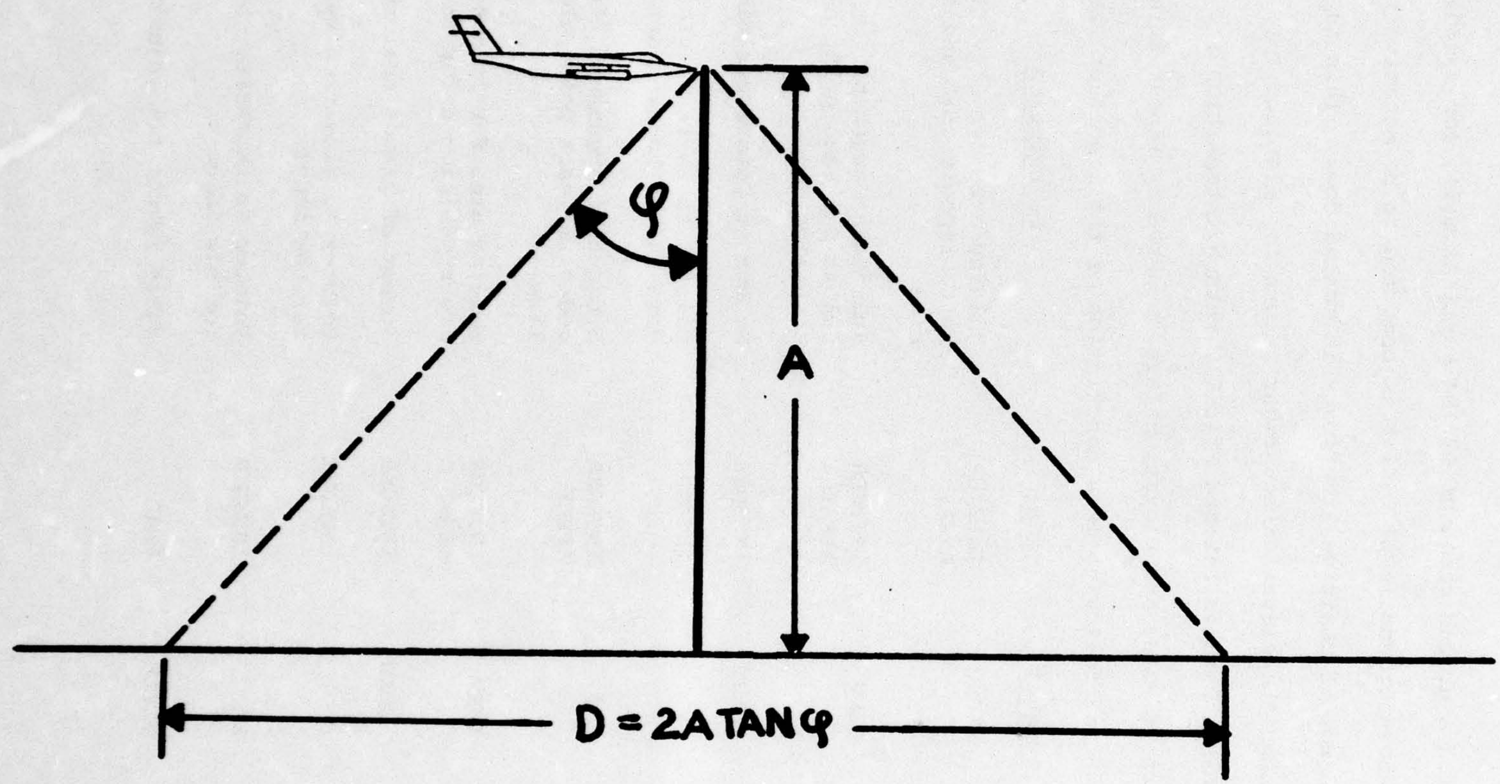


FIGURE II.20– RADAR GROUND COVERAGE AS A FUNCTION OF ALTITUDE AND APERTURE

covering a ground range of 19.194 nautical miles for a radar located 32000 feet above ground level. If F is computed to be negative the image is scaled up; if positive, the image is scaled down. This is done simply by assigning the elevation and radar return for distance r to distance F*r. For points between F*r and F*(r+1), return intensities are set equal to the background value and elevations are computed by linear interpolation.

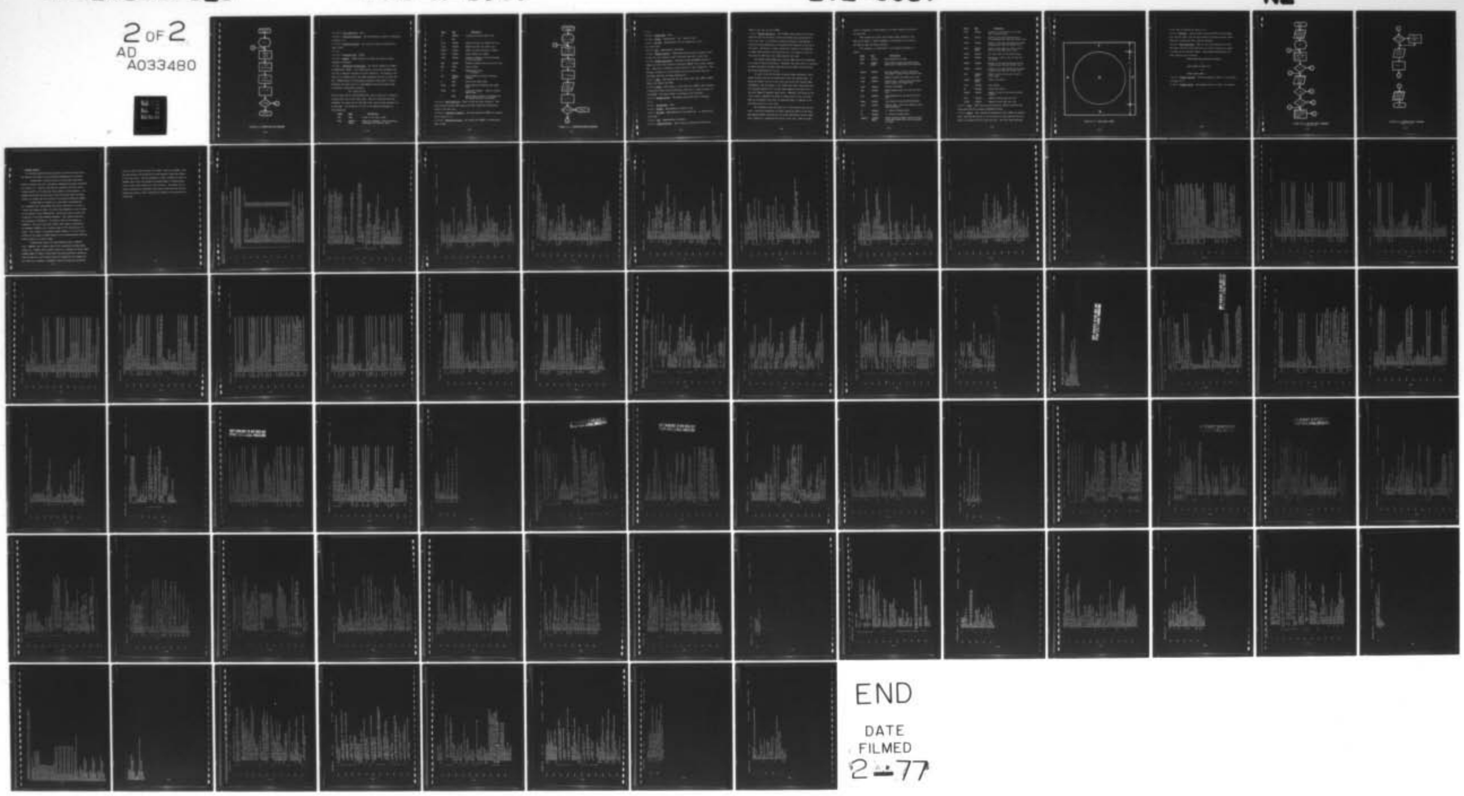
The following is a description of the important variables in SIZE.

Label	Type	Description
ELESV	INTEGER ARRAY	Holding area for elevation values that are being repositioned and scaled to new image size
ELEV	INTEGER ARRAY	Subroutine argument. Array of elevation values for each point along a radial scan line
IDIST	INTEGER	Number of points per radial scan line
IE	INTEGER	Pointer to where to place scaled image data in the holding areas
INS	INTEGER ARRAY	Subroutine argument. Array of intensity codes for each point along a radial scan line
INSV	INTEGER ARRAY	Holding area for intensity values that are repositioned for the new image size
ISHADE	INTEGER	Number of pixels assigned to the shadow
J	INTEGER	Pointer to elevation value to be scaled for new image
KK	INTEGER	Pointer to intensity code to be retrieved for new image
SCALE	REAL	Scale factor for adjusting the image size

AD-A033 480   PRC INFORMATION SCIENCES CO MCLEAN VA   F/G 17/9
PROGRAM MAINTENANCE MANUAL FOR THE REFERENCE SCENE SOFTWARE (RS--ETC(U)
OCT 76   C B SHELTON   DAAK02-75-C-0098

UNCLASSIFIED   PRC-R-1939   ETL-0067   NL

2 OF 2
AD
A033480



END
DATE
FILMED
2-77

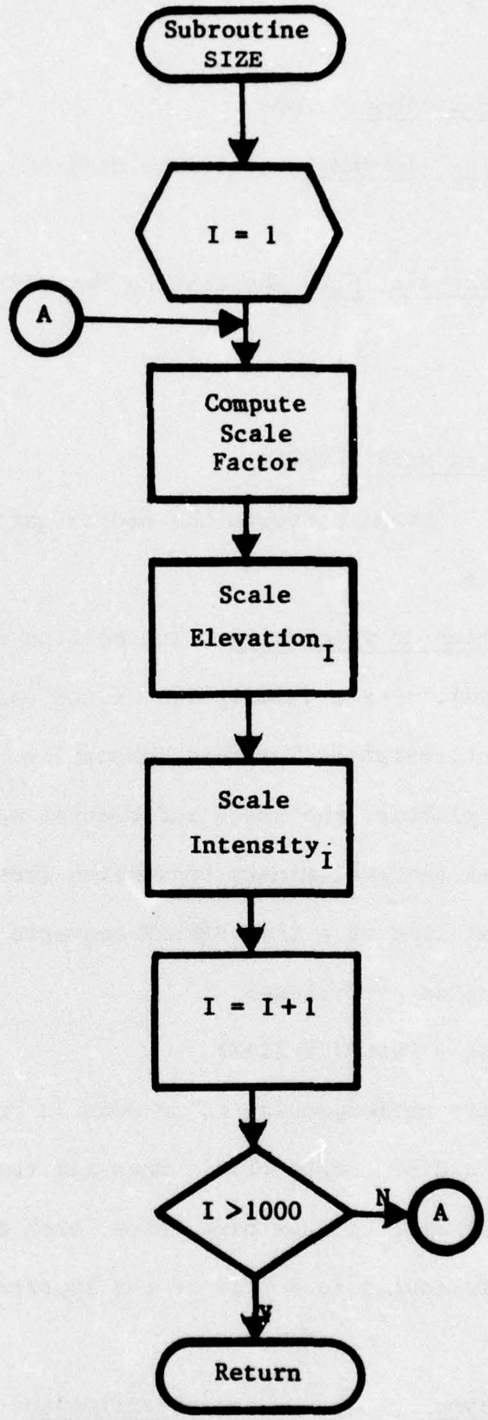


FIGURE II.21 - SUBROUTINE SIZE FLOWCHART

2.2.9.15.4 Error Conditions. None

2.2.9.15.5 Subroutine Flowchart. The flow diagram for SIZE is illustrated in Figure II.21.

2.2.9.15.6 Subroutine Listing. The listing for SIZE is included with that of RSS9.

2.2.9.16

2.2.9.16.1 Subroutine Name. REFRMT

2.2.9.16.2 Summary. REFRMT converts the radial data back to raster cartesian coordinates.

2.2.9.16.3 Description of Processing. The calling sequence for REFRMT is REFRMT(DATA, THETA). For a description of the calling arguments see the list of important variables for this subroutine. For purpose of display on the DICOMED plotter, the image information must be in raster format. REFRMT performs the preliminary conversion from radial to raster. Processing one radial line at a time REFRMT converts each radial point to cartesian coordinates as follows:

$$YX = 2000*((Y-1)+X)$$

This YX value with its corresponding color code is output as a record for a single point into a disc output file. When all the points have been processed, the result will be four disk files, each of which contains 1/4 of the image. The following is a list of the important variables of this subroutine.

Label	Type	Description
CTH	REAL	Cosine of the radial angle.
DATA	INTEGER ARRAY	Subroutine argument. Array containing elevations and intensity codes.

Label	Type	Description
ID	INTEGER	Current point along radial line.
D	REAL	" " " " "
IDIST	INTEGER	Number of points per radial scan line.
ILINE	INTEGER	Number of points per raster line.
ISHADE	INTEGER	Number of pixels assigned to shadow.
IUNIT	INTEGER	Output device unit number.
KOUNT	INTEGER	Counter of number of points processed by this subroutine.
NMAX	INTEGER	Maximum allowable raster points per line.
NPIS	INTER	Same as IDIST.
PI	REAL	Mathematical value. = 3.1515928
OUT	INTEGER ARRAY	Output buffer for raster converted records.
STH	REAL	Sine of the radial angle
SCALE	REAL	Scale factor for adjusting the image size.
THETA	REAL	Subroutine argument. Angle of current radial line.
Y	REAL	Random number between 0-1 generated by the random function routine RANF.

2.2.9.16.4 Error Conditions. There is only one error condition. This error condition occurs when there is an error found while outputting a record to the disk file.

2.2.9.16.5 Subroutine Flowchart. The flow diagram for REFRMT is illustrated in Figure II. 22.

2.2.9.16.6 Subroutine Listing. The listing for REFRMT is included with that of RSS9.

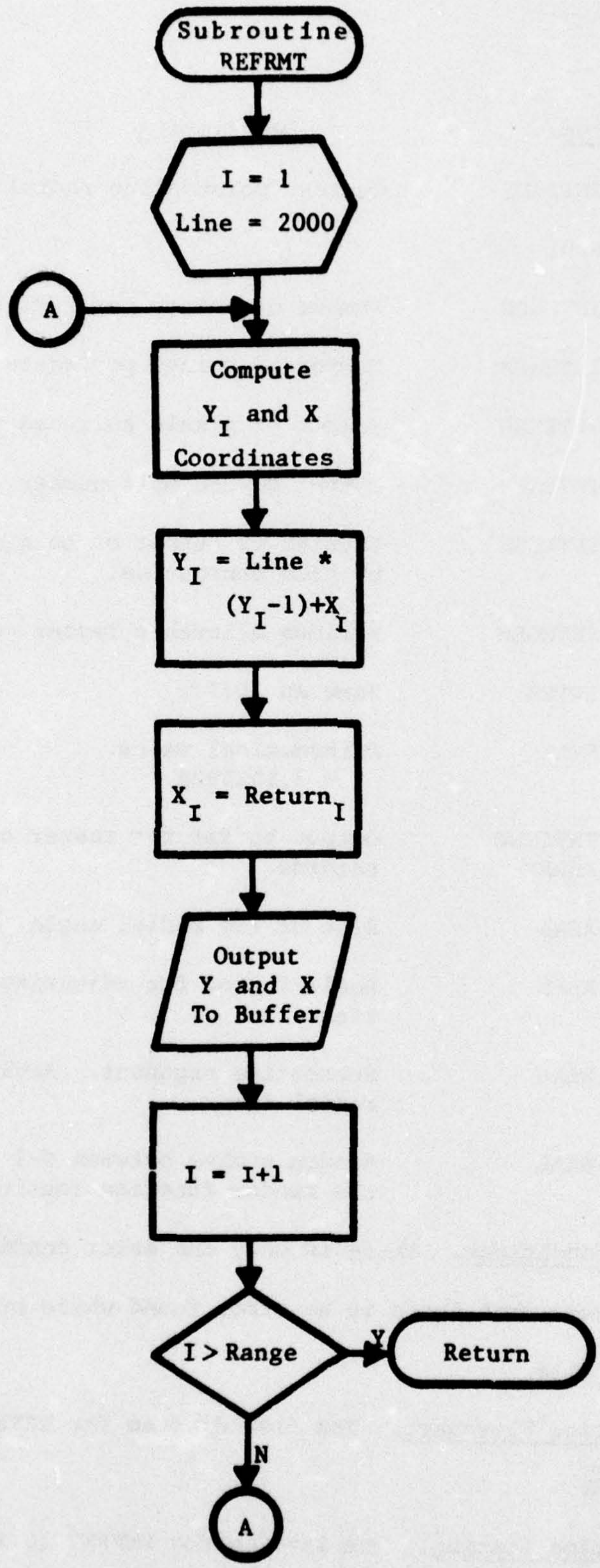


FIGURE II.22 - SUBROUTINE REFRMT FLOWCHART

2.2.10

2.2.10.1 Program Name. SORT

2.2.10.2 Storage. Approximately 130K words of core.

2.2.10.3 Run Time. Approximately 1750 cp seconds and 2 hours wall clock time.

2.2.10.4 Cost. Approximately 350 dollars.

2.2.10.5 Program Function. Program SORT sorts the point records on each of the four files output by RSS9 and merges them into a single file.

2.2.10.6 Program Description. SORT uses the CDC SORT/MERGE package to combine the four files, output by subroutine REFRMT, into a single ordered file. This file is ordered by a mod function of Y and X (MOD(Y,C)). The procedure followed is to individually sort each of the four files and then merge them into one large ordered file.

2.2.10.7 Input. SORT requires the four output disk files (TAPE3, TAPE10, TAPE12, and TAPE14) from RSS9.

2.2.10.8 Output. SORT outputs a single disk file (TAPE27) which contains 1.44 million ordered records describing each point to later be output to the DICOMED. The record format is the same as that from RSS9.

2.2.10.9 Program Listing. The program listing for SORT is attached.

2.2.11

2.2.11.1 Program Name. RSS10

2.2.11.2 Storage. Approximately 3K words of core.

2.2.11.3 Run Time. Approximately 510 cp seconds and 20 minutes wall clock time.

2.2.11.4 Cost. Approximately 65 dollars.

2.2.11.5 Program Function. RSS10 uses the sorted data from SORT to

format a plot tape for the DICOMED.

2.2.11.6 Program Description. The DICOMED Screen consists of a 2048 x 2048 grid with the intensity of each grid element being specified by a 6-bit color code. In the mode of operation presently being employed, plotting is done sequentially in horizontal rows beginning at the top of the screen. Therefore, in order to describe a picture it is necessary to format 2048 records of 205 words each e.g., one record for each row consisting of 2048 6-bit color codes packed 10 per word.

The present image format calls for the radar scene to be displayed as a circle of radius 1000 pixels. Therefore, the first step is to generate 11 blank (all white) records to describe the top margin of the picture. The actual image begins in row 12.

For each of the 2000 records containing image information, those pixels actually lying in the radar-scene circle are colored black. As should be evident, not all points within the circle will contain image information - only the points on the radial scan lines contain data and if an angular spacing of $\frac{1}{2}^{\circ}$ is used, these comprise only about 1/4 of the total number of possible image points. Therefore, the choice of the "fill" color is important and black is chosen since it will not introduce any correlation error when the generated image is compared to the more dense output of a real radar.

The information from the input file is then written over the fill color. Since the information is sorted, taking the points in the order they appear permits the records to be filled sequentially left to right, with a change in Y signalling the end of a given line. When the radar

circle is completed, a bottom margin of 12 blank records is written to the plot tape.

This program also places four fiducial marks located at each edge of the image. A small crossmark is also placed at the center of the image to mark the target location.

The following is a description of the important variables in RIMS10.

Label	Type	Description
BACK6	INTEGER	The background fill code.
DATA	INTEGER ARRAY	Input buffer record containing information for a single point to be plotted.
IDELTA	INTEGER	Half the number of points covered by the bounds of radar circle for this line.
IDIST	INTEGER	Number of points per radial scan line.
IFLAG	INTEGER	Control flag. Value of this flag determines location of program jump.
ILINE	INTEGER	Current line number.
IMAX	INTEGER	Right boundary point of circle for this line.
IMIN	INTEGER	Left boundary point of circle for this line.
INDEX	INTEGER	Line index for top and bottom margins.
IPASS	INTEGER	Control flag. Counts the number of data lines processed.
IX	INTEGER	X - value of current point.
IY	INTEGER	Y - value of current point.
JBSHIFT	INTEGER ARRAY	Table containing number of bits to shift 6-bit color code into a word containing 10 codes.

Label	Type	Description
JBYTE	INTEGER	Pointer to a 6-bit byte in a 10 byte word of output buffer.
JBYTE1	INTEGER	Pointer to the byte containing the left boundary point data for current line.
JBYTE2	INTEGER	Pointer to the byte containing the right boundary point data for current line.
JMASK	INTEGER ARRAY	Array of 6-bit masks to insert each of the ten color codes into a word.
JPIXEL	INTEGER	The color code for the current point.
JWORD	INTEGER	Pointer to a word in the 205 word output buffer.
JWORD1	INTEGER	Pointer to the word containing the left boundary point data for current line.
JWORD2	INTEGER	Pointer to the word containing the right boundary point data for current line.
LBUF	INTEGER ARRAY	Array of counters for each of the 64 color codes.
LINBUF	INTEGER ARRAY	Output line buffer.
LINE	INTEGER	Line counter.
MP	INTEGER	Output file parity.
NBDRLN	INTEGER	Number of lines in the top and bottom margins.
NPTS	INTEGER	Number of points per scan line.
PICSIZ	INTEGER	Number of points per scan line.

2.2.11.7 Input. RSS10 has one input disk file (TAPE1) containing the sorted data from the preceding SORT/MERGE program.

2.2.11.8 Output. RSS2 outputs the DICOMED plot file (TAPE3) on magnetic tape. Each 205 word record of the file contains color code data for one raster line packed ten 6-bit codes per word. The final image generated

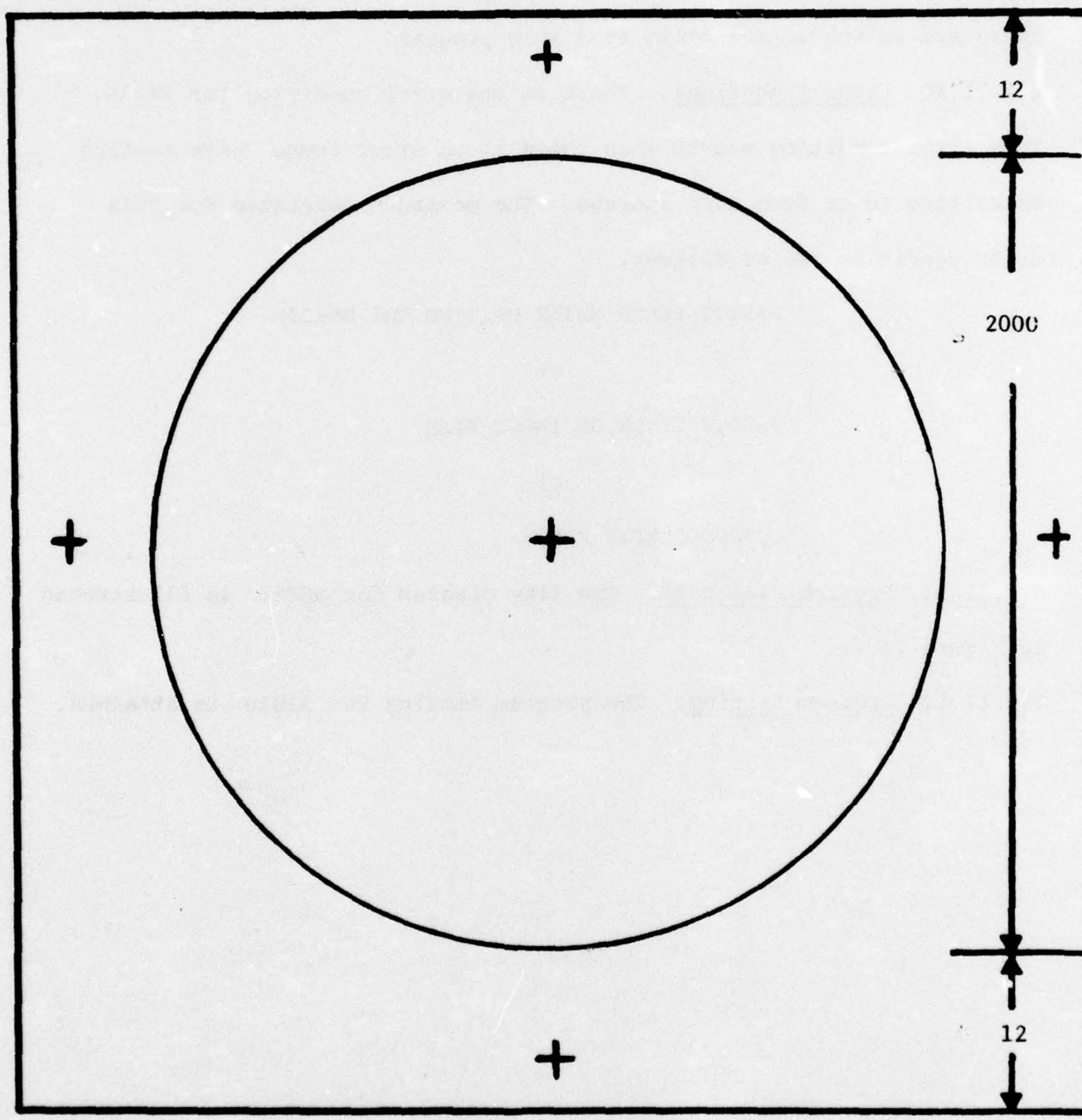


FIGURE II. 23- FINAL IMAGE FORMAT

by the tape is presented in Figure II. 23.

2.2.11.9 Externals. System routines called by RSS10 are the square root function (SQRT); and the mass storage I/o function routine (UNIT). There are no subroutine calls from this program.

2.2.11.10 Error Conditions. There is one error condition for RSS10. This error condition occurs when there is an error found while reading or writing to or from disk storage. The messages generated for this error condition are as follows.

PARITY ERROR WHILE WRITING TOP MARGIN

or

PARITY ERROR ON INPUT FILE

or

OUTPUT PARITY ERROR.

2.2.11.11 Program Flowchart. The flow diagram for RSS10 is illustrated in Figure II. 24.

2.2.11.12 Program Listing. The program listing for RSS10 is attached.

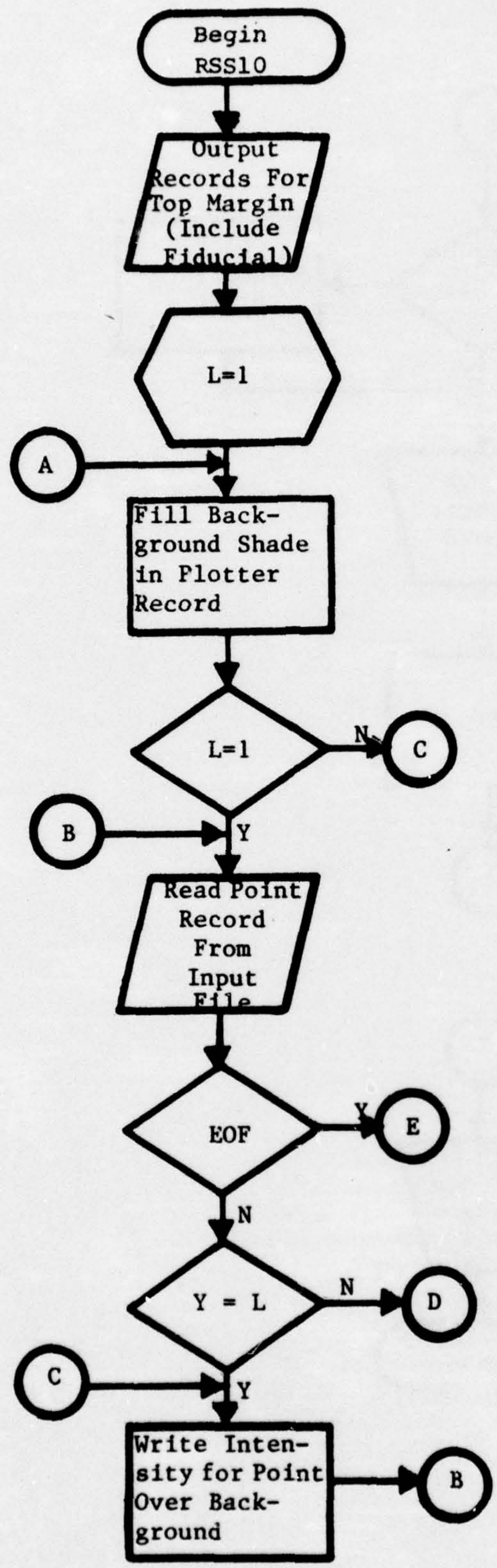


FIGURE II.24 - PROGRAM RSS10 FLOWCHART
(Page 1 of 2)

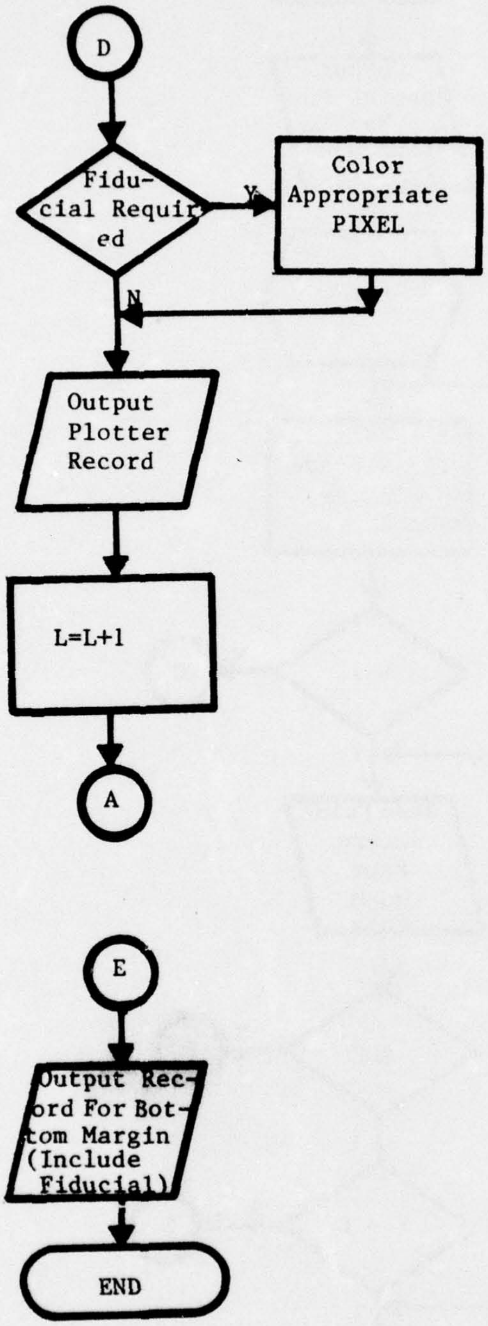


FIGURE II.24 - PROGRAM RSS10 FLOWCHART
(Page 2 of 2)

3. <u>Program Listings</u>

The enclosed program listings represent the latest version of the RSS software and differ from the preceding documentation as follows:

Program RSS8, in present form, will skip every other point along the radial lines that it generates, degrading the radial resolution by a factor of 2. This was done when the azimuthal resolution was increased from $\frac{1}{2}^{\circ}$ to $\frac{1}{4}^{\circ}$ to keep the total quantity of data constant. The program also prints out the X and Y errors that result when the target location is rounded off to the center of the nearest resolution element.

Program RSS9 now accepts up to four input card parameters, via a NAMELIST read, and degrades the radial resolution by a factor of 2 as per the changes to RSS8. The first input parameter is the altitude of the reference scene, REFALT=32000. The altitude layover effect will be applied if the input parameter LAYOVER=0. The layover effect will not be applied if LAYOVER ±0. The default value in the program is LAYOVER=0. The size of the final output raster image is specified by the parameter IRASTER, with a default value of 301 (301 pixels by 301 lines). And, finally, the parameter ZANGLE (default = 0.0) will allow rotation of the image by ZANGLE degrees in the counterclockwise direction before conversion to raster format.

Program RSS10 inputs five card parameters with a NAMELIST read. IRASTER, with a default value of 301, specifies the input raster image size. IBLOWUP, with a default value of 5, enlarges the raster image IBLOWUP number of times to better match the output plotter's resolution. IOFX and IOFY are X and Y offsets that will reposition the enlarged output raster by a maximum of ± IBLOWUP/2 times, and these are calculated

from the X and Y errors printed out by RSS8. The use of IBLOWUP, IOFX, and IOFY allows a low resolution but high geometric centering accuracy of the final scene. The last parameter is ISKIP, normally set equal to IRASTER, which tells the program the maximum number of skipped pixels along a final raster output line to be filled in. The radial line to raster conversion will sometimes leave raster pixels unfilled, and the program will place in these locations the average of the intensities on either side.

```
      PROGRAM  RSS1(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE7=200,
     .TAPE8=0)
C
C*********************************************************************
C*********************************************************************
C     UNBLOCK DATA,CORRECT ERRORS AND FILL RIVERS
C*********************************************************************
C*********************************************************************
C
C
C     RADAR GREY SHADES                                  00001 THRU 00020
C     LARGE RIVERS (WATER ON LEFT)                       10110
C     LARGE RIVERS (WATER ON RIGHT)                      10120
C     DAMS                                               10130
C     MARSHES AND SWAMPS                                 10140
C     LAKES                                              10150
C     ISLANDS                                            10160
C     RIVERS AND STREAMS                                 10170
C     RAILROAD YARDS                                     10210
C     RAILROADS                                          10220
C     TOWNS AND SUBURBS                                  10310
C     MEDIUM CITIES AND COMMERCIAL AREAS                 10320
C     BIG CITIES AND INDUSTRIAL AREAS                    10330
C     LARGE ISOLATED BUILDINGS                           10340
C     INTERSTATE HIGHWAYS AND TURNPIKES                  10410
C     MAJOR ROADS                                        10420
C     SECONDARY ROADS                                    10430
C     UNPAVED ROADS AND TRAILS                           10440
C     AIRPORT                                            10450
C     POWER LINE TOWERS (WITH CABLES)                    10510
C     DRIVE-IN MOVIES                                    10520
C     FIRE OR RADIO TOWERS                               10530
C     CEMETERIES                                         10540
C     POL AREA                                           10550
C     HARDWOOD FOREST                                    10610
C     EVERGREEN FOREST                                   10620
C     MEADOWS AND GRASSY FIELDS                          10630
C     DRY ROCKY AREAS                                    10640
C     SAND AND SAGEBRUSH AREAS                           10650
C     SNOW COVERED AREAS                                 10660
C     DRY RIVERBEDS, CANALS, AND STORM DRAINS            10670
C     DRY LAKE BEDS AND GULCHES                          10680
C     RIVER FILL                                         10710
C
      DIMENSION X2(20000),X3(10000),X4(10000),NOCLOSE(15)
      INTEGER BLOCK,START,RESTART,FINISH,UP,DOWN,YES,PEN,POINT,BLANK,
     .HEADER,FTLIF(32),BUFFER(200),TYPE(34),OUTAP
C
      DATA INMASK/773/,INTAP/7/,OUTAP/8/
      DATA START/51B/,YES/'YES'/,NO/'NO'/,POINT/57B/,FINISH/02B/,
     .UP/25B/,DOWN/04B/,BLANK/55B/,MINUS/46B/
      DATA NOCLOSE/10110,10120,10130,10170,10220,10410,10420,10430,
     .10440,10510,10520,10670,10670,10340,10530/
```

```
      DATA ITLIF/10510,10520,10530,10310,10210,10130,10320,10540,10420,
     .10450,10340,10140,10150,10160,10220,10170,10330,10410,10430,
     .10440,10110,10120,10710,10550,10610,10620,10630,10640,10650,
     .10660,10670,10680/
      DATA TYPE/'POWERTOWER','DRIVE-IN','FIRE TOWER','TOWN','RAILYARD',
     .'DAM','MED. CITY','CEMETERY','MAJOR ROAD','AIRPORT','BUILDING',
     .'SWAMP','LAKE','ISLAND','RAILROAD','SML. RIVER','BIG CITY',
     .'HIGHWAY','SEC. ROAD','TRAIL','RIGHT BANK','LEFT BANK',
     .'RIVER FILL','POL AREA','HARDFOREST', 'EVGRFOREST' ,'MEADOW',
     .'ROCKY AREA','SAND','SNOW','CANAL','DRY LAKE','POWER LINE',
     .'RADARSHADE'/
C
      NAMELIST / PARAMS / XORGIN,YORGIN
      XORGIN=YORGIN=0.0
      READ PARAMS
      WRITE(6,2)
    2 FORMAT(' SEQUENCE NO.',7X,'FEATURE CODE',7X,'DESCRIPTION',7X,
     .'CLOSURE',7X,'NO. OF POINTS',7X,'MINX',7X,'MAXX',7X,'MINY',7X,
     .'MAXY'/)
C
C
C        SPECIAL EDITING FOR TARGET 4 ONLY. FIRST SET (23 OF THEM) 10999 CODES
C        ARE IGNORED. SECOND SET ARE CHANGED TO LAKE CODES (RIVERS ARE A SERIES
C        OF CONNECTED LAKES). ALSO, FIRST TWO 10130 CODES ARE DAMS, BUT REST
C        ARE RAILROADS.
      IZRIV=IZRAIL=0
C
C****************************
C     READ IN CORRECTED ORIGINS FOR X AND Y COORDINATES
C     BEGIN NEXT FEATURE RECORD OR CONTINUE OLD FEATURE
C     RECORD TO NEXT DATA BLOCK
C
C     CHECK FOR FORMAT ERRORS
C****************************
      NEWBLK=YES$RESTART=NO$HEADER=NO$IF=0$IWD=180
C     KFLAG=1 $ISEQ=0 $IBYTE=6 $LOC=15 THIS WAS FOR 90 MILE TARGET
      KFLAG=1 $ISEQ=0 $IBYTE=1 $LOC=91
      JZ=0
  100 IF(JZ.NE.0) GO TO 440
C 100 IF(NEWBLK.NE.YES) GO TO 103
      IF(NEWBLK.NE.YES) GO TO 103
      BLOCK=BLOCK+1
      BUFFER IN(INTAP,0)(BUFFER(1),BUFFER(200))
      IND1=UNIT(INTAP)
      IWD=LENGTH(INTAP)
      NEWBLK=NO
      IF(UNIT(INTAP)) 103,300,301
  103 ISHIFT=60-6*IBYTE
      ITEMP=AND(SHIFT(BUFFER(LOC),-ISHIFT),TMASK)
      IF(ITEMP.EQ.START.OR.ITEMP.EQ.67B) GO TO 107
      IF(ITEMP.NE.FINISH) GO TO 106
      NEWBLK=YES
      LOC=1$IBYTE=1$GO TO 100
```

```
  106 IF(RESTART.EQ.YES) GO TO 105
      IF(HEADER.EQ.NO) GO TO 500
      IF(IF.NE.0) GO TO 112
      GO TO 109
C*****************************
C     LOCATE FEATURE CODE
C*****************************
  107 HEADER=YES
      ICLOSE=NO
      XMIN=99.999 $XMAX=0. $YMAX=0. $YMIN=99.999
      IF=0
      LOC=LOC+1
      IF(IBYTE.GE.8) LOC=LOC+1
      IBYTE=IBYTE+3
      IF(IBYTE.GT.10) IBYTE=IBYTE-10
      IF(LOC.LE.IWD) GO TO 109
      NEWBLK=YES
      LOC=1$GO TO 100
C**************************
C     FEATURE CODE BEGINS IN BYTE 13 OF RECORD. THIS MAY BE BEYOND THE
C     END OF THE PRESENT BLOCK. CHECK FOR THIS. IF SO, DETERMINE LOCATION
C     OF FEATURE CODE IN NEXT BLOCK AND READ IN NEXT BLOCK
C**************************
  109 ISHIFT=60-6*IBYTE
      ITEMP=AND(SHIFT(BUFFER(LOC),-ISHIFT),IMASK)
      IF(ITEMP.NE.FINISH) GO TO 101
      KOUNT=0
      NEWBLK=YES
   50 IBYTE=IBYTE-1
      KOUNT=KOUNT+1
      IF(IBYTE.EQ.0) LOC=LOC-1
      IF(IBYTE.EQ.0) IBYTE=10
      ISHIFT=60-6*IBYTE
      ITEMP=AND(SHIFT(BUFFER(LOC),-ISHIFT),IMASK)
      IF(ITEMP.EQ.FINISH) GO TO 50
      IF(KOUNT.GT.10) GO TO 51
      LOC=1$IBYTE=KOUNT$GO TO 100
   51 LOC=2$IBYTE=KOUNT-10$GO TO 100
C****************************
C     CONSTRUCT FEATURE CODE
C****************************
  101 IEXP=4
      IBYTE=IBYTE-1
      DO 108 K=1,5
      IBYTE=IBYTE+1
      IF(IBYTE.LE.10) GO TO 102
      IBYTE=1
      LOC=LOC+1
  102 ISHIFT=60-6*IBYTE
      ITEMP=AND(SHIFT(BUFFER(LOC),-ISHIFT),IMASK)
      IF=IF+(ITEMP-27)*(10**IEXP)
  108 IEXP=IEXP-1
C
```

```
      C        FOR HUNTSVILLE TARGETS  ONLY. RIVER CODES INCORRECT. WILL IGNORE
      C        THEM. WILL DIVIDE RIVER UP INTO A SERIES OF LAKES AND THESE
      C        ARE CODED ON THE DIGITIZER TAPE AS FEATURE CODE 10999.
            IF(IF.EQ.10130) IZRAIL=IZRAIL+1
            IF((IF.EQ.10130).AND.(IZRAIL.GT.2)) IF=10220
            IF(IF.EQ.10999) IZRIV=IZRIV+1
            IF((IF.EQ.10999).AND.(IZRIV.GT.20)) IF=10140
      C
        408 IBYTE=IBYTE+1
            IF(IBYTE.GT.10) LOC=LOC+1
            IF(IBYTE.GT.10) IBYTE=1
            IF(LOC.LE.TWO) GO TO 112
            NEWBLK=YES
            LOC=1$GO TO 100
      C***************************
      C     CHECK PEN COMMAND ... SHOULD BE PEN=DOWN
      C***************************
        112 ISHIFT=60-6*IBYTE
            PEN  =AND(SHIFT(BUFFER(LOC),-ISHIFT),IMASK)
            IF(PEN.NE.DOWN.AND.PEN.NE.FINISH) GO TO 505
            IF(PEN.NE.FINISH) GO TO 119 .
            NEWBLK=YES
            IBYTE=1$LOC=1$GO TO 100
        119 IBYTE=IBYTE+1
            IF(IBYTE.GT.10) LOC=LOC+1
            IF(IBYTE.GT.10) IBYTE=1
            HEADER=NO
            RESTART=YES
            IN=0
            IZCLOSE=+13 $ JZ=0
            DO 420 I=1.15
            IF(IF.EQ.NOCLOSE(I)) IZCLOSE=0
        420 CONTINUE
            IF(LOC.LE.TWO) GO TO 105
            NEWBLK=YES
            LOC=1$IBYTE=1$GO TO 100
      C****************************
      C     PULL OFF (X,Y) COORDINATES
      C****************************
        105 IEXP=2
            TEMP=0. $ I=LOC $ ZSIGN=+1.
      C     DO 112 I=LOC,TWO
        400 NWORD=I
      C     DO 111 J=IBYTE,10
            J=IBYTE
        410 NBYTE=J
            ISHIFT=60-6*J
            ITEMP=AND(SHIFT(BUFFER(I),-ISHIFT),IMASK)
            IF(ITEMP.EQ.46B) ZSIGN=-1.
            IF((ITEMP.EQ.BLANK).OR.(ITEMP.EQ.463)) GO TO 113
            IF((ITEMP.EQ.UP).OR.(ITEMP.EQ.START)) GO TO 115
            IF(ITEMP.EQ.FINISH) GO TO 116
            IF(ITEMP.EQ.POINT) GO TO 114
```

```
      ITEMP=ITEMP-27
      TEMP=FLOAT(ITEMP)*(10.**IEXP)+TEMP
  113 IEXP=IEXP-1
  114 IF(IEXP.GT.-4) GO TO 111
      IF(IN.LT.2000) GO TO 430
      IF(IZCLOSE.NE.0) GO TO 430
      ZTEMP=TEMP $ NWORDZ=NWORD $ NBYTEZ=NBYTE
      JZ=13 $ IZF=IF $ JI=I $ JJ=J $ GO TO 94
  440 I=JI $ J=JJ $ IN=0 $ IF=IZF $ JZ=0
      TEMP=ZTEMP $ NWORD=NWORDZ $ NBYTE=NBYTEZ
      XMIN=99.999 $ XMAX=0.0 $ YMIN=99.999 $ YMAX=0.0
  430 IN=IN+1
      X2(IN+3)=TEMP*ZSIGN
C**************************
C     APPLY CORRECTION FACTOR TO Y COORDINATES. IF Y.LT.0 DISCARD POINT
C     DETERMINE COORDINATE BOUNDS OF FEATURE
C**************************
      IF(MOD(IN,2).EQ.1) GO TO 104
      X2(IN+3)=X2(IN+3)-YORGIN
      IF((X2(IN+2).GT.0.).AND.(X2(IN+3).GT.0.)) GO TO 460
      IN=IN-2
      IF(IN.LT.0) IN=0
      GO TO 123
  460 IF(X2(IN+3).GT.YMAX) YMAX=X2(IN+3)
      IF(X2(IN+3).LT.YMIN) YMIN=X2(IN+3)
      GO TO 123
  104 X2(IN+3)=X2(IN+3)-XORGIN
      IF(X2(IN+3).LT.0.) GO TO 123
      IF(X2(IN+3).LT.XMIN) XMIN=X2(IN+3)
      IF(X2(IN+3).GT.XMAX) XMAX=X2(IN+3)
  123 TEMP=0. $ IEXP=2 $ ZSIGN=+1.
C 111 CONTINUE
  111 J=J+1
      IF(J.LE.10) GO TO 410
      IBYTE=1
C 110 CONTINUE
  110 I=I+1
      IF(I.LE.IWD) GO TO 400
  116 NEWBLK=YES$LOC=1$IBYTE=1$GO TO 100
C***************************
C     FEATURE RECORD IS COMPLETE, NOW PROCESS
C***************************
  115 RESTART=NO
      LOC=NWORD
      IBYTE=NBYTE+1
C
C        DO NOT GO TO NEXT BYTE IF DIGITIZER SCREWED UP AND FORGOT TO
C        PUT IN THE U FOR PEN UP (PUT IN A ( INSTEAD)
      IF(ITEMP.EQ.START) IBYTE=NBYTE
      IF(IBYTE.LE.10) GO TO 117
      LOC=LOC+1$IBYTE=1
C**************************
C     CHECK TO SEE IF NEXT FEATURE BEGINS IN A NEW DATA BLOCK
```

```
C***************************
  117 ISHIFT=60-6*IBYTE
      ITEMP=AND(SHIFT(BUFFER(LOC),-ISHIFT),IMASK)
      IF(ITEMP.NE.FINISH.AND.LOC.LE.IND) GO TO 94
      LOC=1$IBYTE=1$NEWBLK=YES
C****************************
C     CHECK FOR VALID FEATURE CODE
C     LOWEST FEATURE CODE FOR RADAR GREY SHADES IS 20101
C****************************
C
C          DO NOT USE CORNER TICK MARKS (CONTROL POINTS)
   94 IF(IF.EQ.10700) GO TO 100
C
C          EDIT FOR THIS TARGET
      IF(IF.EQ.10999) GO TO 100
C
  194 IF(IF.GE.10110.AND.IF.LE.10710) GO TO 93
      IF(IF.GE.20101.AND.IF.LE.20420) GO TO 93
      GO TO 100
C****************************
C     CHECK FOR CLOSURE AND CLOSE IF NECESSARY
C
C     FEATURES WITH THE TOPOLOGY OF ROADS ARE NOT CLOSED
C****************************
   93 DO 120 I=1,15
      IF(IF.EQ.NOCLOSE(I)) GO TO 121
  120 CONTINUE
      IF(IF.GE.20101.AND.IF.LE.20320) GO TO 121
      IF(IN.LE.2) GO TO 100
      ICLOSE=YES
      X2(IN+4)=X2(4)
      X2(IN+5)=X2(5)
      IN=IN+2
  121 CONTINUE
C***************************
C     CONVERT ETL FEATURE CODES TO NAVY CODES
C     WHERE NO CORRESPONDENCE EXISTS, REDUCE ETL CODES TO VALUES LESS THAN 53
C     DO NOT CONVERT ETL CODES PERTAINING TO RADAR GREY SHADES
C***************************
      DO 122 I=1,32
      IF(ETLIF(I).NE.IF) GO TO 122
      IF=I
      GO TO 7
  122 CONTINUE
C****************************
C     INSERT FEATURE RECORD SUMMARY INTO X2 ARRAY
C****************************
    7 ISEQ=ISEQ+1
      X2(1)=ISEQ
      X2(2)=IF
      IF(IF.EQ.3.OR.IF.EQ.11) IN=2
      X2(3)=IN
      KMAX=IN+3
```

```
   30 IF(IF.NE.1) GO TO 31
      IF=3$X2(2)=IF
C****************************
C     WRITE TO TAPE + LIST ON PRINTER
C****************************
C***************************
C     IF FEATURE IS A POWER LINE, SEPARATE THE TOWER FROM THE CABLE
C     AND WRITE EACH AS A SEPARATE FEATURE
C***************************
   31 N=IN/2
      IF(IN.LE.0) GO TO 9
      IF(IF.LE.33) ITYPE=TYPE(IF)
      IF(IF.GT.33) ITYPE=TYPE(34)
      WRITE(6, 3) ISEQ,IF,ITYPE,ICLOSE,N,XMIN,XMAX,YMIN,YMAX
    3 FORMAT(4X,I4,15X,I5,12X,A10,9X,A3,13X,I4,10X,3(F7.3,4X),F7.3)
   10 BUFFER OUT (OUTAP,1) (X2(1),X2(KMAX))
      INDIC=UNIT(OUTAP)
      NWD=LENGTH(OUTAP)
      IF(INDIC) 9,199,199
    9 CONTINUE
      IF(IF.NE.33) GO TO 15
      IN=2
      IF=1$X2(2)=IF$X2(3)=2$KMAX=5$ISEQ=ISEQ+1$X2(1)=ISEQ$GO TO 10
C****************************
C     FILL RIVERS
C
C     RIGHT BANK OF RIVER
C****************************
   15 IF(IF.NE.21) GO TO 45
      IN3=IN
      GO TO 12
C
C     LEFT BANK OF RIVER
C
   45 IF(IF.NE.22) GO TO 100
      IN4=IN
      GO TO 16
C
C     PLACE RIGHT BANK DATA POINTS IN ARRAY X3
C
   12 INP2=IN+2
      DO 14 I=4,INP2,2
      X3(I)=X2(IN-I+6)
      X3(I+1)=X2(IN-I+7)
   14 CONTINUE
      GO TO 18
C
C     PLACE LEFT POINTS IN ARRAY X4
C
   16 INP3=IN+3
      DO 17 I=4,INP3
   17 X4(I)=X2(I)
   18 CONTINUE
```

```
      IF(KFLAG.NE.1) GO TO 19
      KFLAG=2
      GO TO 100
   19 IF=23
      ICLOSE=YES
      IT=IN4+IN3+2
C
C     CLOSE RIVERS INTO POLYGONS
C
      IN4P3=IN4+3
      DO 25 I=4,IN4P3
   25 X2(I)=X4(I)
      IN3P3=IN3+3
      DO 26 I=4,IN3P3
   26 X2(I+IN4)=X3(I)
      IN=IT
      X2(IN+2)=X4(4)
      X2(IN+3)=X4(5)
      KFLAG=1
      GO TO 7
C****************************
C     WRITE MESSAGE INDICATING RECORD FORMAT ERROR
C****************************
  500 WRITE(6,501) BLOCK $ WRITE(6,511) (BUFFER(I),I=1,IWD)
      IBYTE=IBYTE+1
      IF(IBYTE.LE.10) GO TO 103
      IBYTE=1 $ LOC=LOC+1
      IF(LOC.LE.IWD) GO TO 103
      LOC=1 $ IBYTE=1 $ NEWBLK=YES $ GO TO 100
  501 FORMAT(' INCOMPLETE FEATURE HEADER IN RECORD',I10/)
  505 WRITE(6,506) BLOCK
  506 FORMAT(' ILLEGAL PEN COMMAND IN RECORD',I10/)
  510 WRITE(6,511)(BUFFER(I),I=1,IWD)
  511 FORMAT(30X,'PRESENT RECORD'/10(2X,A10))
      PEN=DOWN $ GO TO 119
C****************************
C     WRITE I/O ERROR MESSAGE
C****************************
  301 WRITE(6,302) BLOCK,IWD,IND1
  302 FORMAT(' I/O ERROR DETECTED AT BLOCK ',I4/' TOTAL WORDS TRANSFERED
     .= ',I3/' BUFFER STATUS= ',I2)
      GO TO 300
  199 WRITE(6,200) BLOCK,NWD,INDIC
  200 FORMAT(' OUTPUT ERROR DETECTED AT BLOCK ',I4/
     .' TOTAL WORDS OUTPUT= ',I3/' BUFFER STATUS= ',I2)
C****************************
C     PRINT SUMMARY
C****************************
  300 PRINT 201,ISEQ
  201 FORMAT(5X,'TOTAL',I4,' FEATURES.')
      END FILE OUTAP
      REWIND OUTAP
      PRINT 1137, XORGIN,YORGIN
```

```
 1137 FORMAT(' DELTA X  AND  DELTA Y  OFFSETS',2F12.5)
      STOP
      END
```

```
      PROGRAM RSS2(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE8=0,
     .TAPE9=0,TAPE20=0)
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C               PROGRAM RLAS2* QUANTIZE TO GRID VALUES
C
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C***********************************************************************
C     THIS PROGRAM QUANTIZES THE EDGES DEFINED BY X-Y PAIRS(VERTICIES)
C     TO GRID VALUES
C***********************************************************************
C***********************************************************************
C     CONVERTS ALL THE TYPES OF DATA, POINTS, LINES AND CONTOURSINTO
C     A 'Y,X,DELTA X,THETA' FORMAT
C***********************************************************************
C***********************************************************************
C     CONTOURS ARE TRAVERSED IN A CW DIRECTION
C***********************************************************************
C***********************************************************************
C     INPUTS TO THIS PROGRAM ARE IN THE FORM OF EDGES, DEFINED BY
C     X-Y POINT PAIRS (VERTICIES)
C***********************************************************************
C***********************************************************************
C     FOR EITHER POSITIVE OR NEGATIVE SLOPE THE Y-X-DELTA X-THETA
C     REPRESENTATION OF THE LINES WILL ALWAYS PROJECT TO THE RIGHT
C     ROUNDOFF HAS BEEN INCLUDED IN THE QUANTIZATION
C***********************************************************************
      DIMENSION X2(7500),IX(15000),JX(15000),KX(7500)
      INTEGER WORD(10)
      INTEGER X5(33)
      INTEGER DISC,INTAP
      INTEGER IN2W
      DATA DISC/3/,INTAP/8/
      DATA IN2W/20/
C***********************************************************************
C     FEATURE TYPE TRANSLATION TABLE
C***********************************************************************
      DATA X5/1,3,1,6,5,3,6,2,4,6,1,6,5,5,3,4,5,4,4,4,3,3,6,6,6,6,5,6,6,
     .6,3,5,3/
C***********************************************************************
C     CONV=6076.11549/156.25 DETERMINES THE SCALE FACTOR FOR THE
C     RESOLUTION ELEMENT OF THE DATA BASE
C***********************************************************************
      NAMELIST / PARAMS / SCALE , IGDSIZ , IPEGSZ
      READ PARAMS
      RESK=IGDSIZ/1000. $ PRINT 60, SCALE,RESK,IPEGSZ $ PRINT 61
   60 FORMAT(//////5X,'SCALE IS 1 TO',F20.5,10X,'RESOLUTION =',F8.3,
     .  '  METERS      WITH',I5,'    CELLS PER REGION'///)
   61 FORMAT(7X,'ISEQ',9X,'IF',9X,'IN',7X,'ISEG',6X,'ISPEC',7X,'ITYP',
     .    8X,'MWD'/)
        RESK=IGDSIZ/304.8
      IF(RESK.LT.14) GO TO 330
      IF(IPEGSZ.GT.50) GO TO 332
```

```
      CONV=(SCALE/12.)/RESK
      PI=ATAN2(0.,-1.)
      IRES=2048
      ISET=0
      ITPC=0
C*************************************************************************
C     READ FEATURE
C*************************************************************************
    1 BUFFER IN (INTAP,1)(X2(1),X2(5000))
      INDIC=UNIT(INTAP)
      NWD=LENGTH(INTAP)
      IF(INDIC.GE.0) GO TO 987
C*************************************************************************
C     SCALE ALL X-Y DATA POINTS FROM THE 0.01 NM TO ELEMENT VALUES
C*************************************************************************
    3 CONTINUE
      DO 4 I=4,NWD
      KX(I-3)=X2(I)*CONV+.5
    4 CONTINUE
      ISEQ=X2(1)
      IF=X2(2)
      IN=X2(3)
C*************************************************************************
C     SPECIAL LOOP TO HANDLE RADAR GREY SHADES
C     THIRD DIGIT OF FEATURE CODE GIVES TOPOLOGY OF FEATURE
C     FIRST TWO DIGITS GIVE GREY SHADE ... FROM 1 TO 20
C     PROCESSED FEATURE CODES ARE IN THE RANGE 41 TO 60
C*************************************************************************
      IF(IF.LT.20101) GO TO 5
      IF=IF-20000
      DO 7 I=2,5
      IF(IF.GT.(100*I)) GO TO 7
      IF=IF-(I-1)*100+40
      ITYP=I+I/5
      ISPEC=0
      IF(I.EQ.3) ISPEC=1
      GO TO 6
    7 CONTINUE
    5 IIF=X5(IF)/2
C*************************************************************************
C     FIX SUMMARY QUANTITIES
C*************************************************************************
      ISPEC=X5(IF)-2*IIF
      ITYP=(X5(IF)+ISPEC)/2
    6 ISEG=0
C*************************************************************************
C     POINT PROCESSING
C*************************************************************************
C*************************************************************************
C     IS THE FEATURE A POINT FEATURE
C*************************************************************************
      IF(ITYP.GT.1)GO TO 110
      INM1=IN-1
```

```
      DO 600 I=1,IMM1,2
      ISEG=ISEG+1
      N=4*ISEG
C*************************************************************************
C     PLACE POINTS IN 'IX' ARRAY IN THE Y-X-DELTA X-THETA FORMAT
C*************************************************************************
      IX(N+3)=KX(I+1)
      IX(N+4)=KX(I)
      IX(N+5)=1
      IX(N+6)=0
  500 CONTINUE
C*************************************************************************
C     SORT POINT FEATURES BY Y,X
C*************************************************************************
      IF(ISEG.EQ.1)GO TO 600
      NDELE=0
      LAST=4*ISEG+3
      LASTM4=LAST-4
      DO 52 K=7,LASTM4,4
      KP4=K+4
      DO 52 KK=KP4,LAST,4
      IF(IX(KK).GT.IX(K))GO TO 52
      IF(IX(KK).LT.IX(K))GO TO 55
      IF(IX(KK+1).GT.IX(K+1))GO TO 52
      IF(IX(KK+1).LT.IX(K+1))GO TO 55
      IX(KK)=99999
      NDELE=NDELE+1
      IX(KK+1)=NDELE
      GO TO 52
   55 DO 552 J=1,4
      ITP=IX(KK+J-1)
      IX(KK+J-1)=IX(K+J-1)
      IX(K+J-1)=ITP
  552 CONTINUE
   52 CONTINUE
      IF(NDELE.GT.0) PRINT 12, ISEQ,NDELE
      IF(NDELE.GT.0)ISEG=ISEG-NDELE
   12 FORMAT(' ISEQ=',I4, '  NDELE=',I4)
C*************************************************************************
C     ADD HEADER INFO + OUTPUT THE WHOLE FEATURE IN THE Y-X-DX-THETA
C     FORMAT
C*************************************************************************
  600 IST=1
      NWD=4*ISEG+6
      WRITE(6,650) ISEQ,IF,IN,ISEG,ISPEC,ITYP,NWD
  650 FORMAT(7(1X,I10))
  715 ISET=ISET+ISEG
      ITRC=ITRC+1
      IX(IST)=ISEQ
      IX(IST+1)=IF
      IX(IST+2)=IN
      IX(IST+3)=ISEG
      IX(IST+4)=ISPEC
```

```
      IX(IST+5)=ITYP
      NWD=4*ISEG+6
  736 MWD=NWD-1800
      IF(NWD.GT.1800)NWD=1800
      BUFFER OUT (DISC,1)  (IX(IST),IX(IST+NWD-1))
      INDIC=UNIT(DISC)
      IF(INDIC.GE.0) GO TO 987
      IF(MWD.LE.0) GO TO 362
      NWD=MWD
      IST=IST+1800
      GO TO 736
  362 CONTINUE
C*********************************************************************
C     IF NOT SPECULAR OR NOT A POLYGON OR FILLED POLYGON GET NEXT
C     FEATURE
C*********************************************************************
      IF(ISPEC.NE.1.OR.ITYP.NE.3.OR.IFILL.LE.IREM)GO TO 1
      ISEG=IFILL-IREM
      ISPEC=0
      IFILL=IREM
      IST=4*IREM+1
      GO TO 715
C*********************************************************************
C*********************************************************************
C     LINE PROCESSING
C*********************************************************************
C     FIND THE CHANGE IN X + Y FOR EACH POINT (VERTEX) PAIR
C*********************************************************************
  110 INM3=IN-3
      DO 250 I=1,INM3,2
      IDX=KX(I+2)-KX(I)
      IDY=KX(I+3)-KX(I+1)
C*********************************************************************
C     DELETE VERTICIES WHICH ARE IDENTICAL
C*********************************************************************
      IF(IDX.EQ.0.AND.IDY.EQ.0) GO TO 250
C*********************************************************************
C     IF FEATURE ISN'T SPECULAR SET THETA=0
C*********************************************************************
      IF(ISPEC.EQ.0)TH=0
      IF(ISPEC.EQ.0)GO TO 106
C*********************************************************************
C     THETA COMPUTATION
C*********************************************************************
C*********************************************************************
C     REVERSE ANGLE FOR WATER ON RIGHT SIDE OF RIVER BANK
C*********************************************************************
      IF(IF.NE.21) GO TO 105
      TH=ATAN2(FLOAT(-IDY),FLOAT(-IDX))
      GO TO 106
  105 TH=ATAN2(FLOAT(IDY),FLOAT(IDX))
C*********************************************************************
C     FIND A SCALED THETA BETWEEN -2048 AND +2047 PLUS A FLAG
```

```
C*************************************************************************
  106 ITH=IRES*(TH/PI)
      IF(ITH.EQ.IRES)ITH=IRES-1
      IF(IDY.GT.0)ITH=ITH-2*IRES
      IF(IDY.LT.0)ITH=ITH+2*IRES
C**** ********************************************************************
C     DY INCREASING IMPLIES LEFT EDGE
C     DY DECREASING IMPLIES RIGHT EDGE
C     ITH POSITIVE FOR LEFT EDGE
C     NSEG COUNTS THE NUMBER OF Y-X-DELTAX-THETA SEGMENTS REQ'D
C**** ********************************************************************
C**** ********************************************************************
C     HORIZONTAL LINE DEFINES ONE LINE SEGMENT
C**** ********************************************************************
      IF(IDY.EQ.0)IDY=1
      NSEG=IABS(IDY)
C**** ********************************************************************
C     FIND SLOPE OF EDGE
C**** ********************************************************************
  107 DX=IDX
      DY=IDY
      RATIO=ABS(DX/DY)
      RATN=0
      IRATN=0
C**** ********************************************************************
C     IRATN IS THE LENGTH OF LINE SEGMENT REQUIRED
C     IRTND IS THE X DISPLACEMENT FROM ORIGINAL VERTIX FOR THE LINE SEG
C**** ********************************************************************
      DO 200 J=1,NSEG
      RATN=RATN+RATIO
      IRTND=IRATN
      IRATN=RATN+.5
      IF(DX*DY.LT.0)IDDX=-IRATN
      IF(DX*DY.LT.0)GO TO 203
      IDDX=IRTND
  203 IDDDX=IRTND-IRATN+1
      IF(IDDDX)201,202,204
  201 IDDDX=-IDDDX
      GO TO 204
C**** ********************************************************************
C     MINIMUM LENGTH SEGMENT IS 1 UNIT LONG
C**** ********************************************************************
  202 IDDDX=1
C**** ********************************************************************
C     FILL THE Y-X-DELTA X-THETA SEGMENTS WITH X + Y INFORMATION
C**** ********************************************************************
  204 IF(IDY.GT.0)JX(4*J-3)=KX(I+1)+J-1
      IF(IDY.GT.0)JX(4*J-2)=KX(I)+IDDX
      IF(IDY.GT.0)GO TO 208
      JX(4*J-3)=KX(I+3)+J-1
      JX(4*J-2)=KX(I+2)+IDDX
C**** ********************************************************************
C     INSERT THE DELTA X + THETA IN THE Y-X-DELTA X-THETA FORMAT ARRAY
```

```
C     JX
C**** ****************************************************************
  206 JX(4*J-1)=IDDRX
      JX(4*J)=ITH
C**** ****************************************************************
C     INCREMENT SUMMARIES
C**** ****************************************************************
  200 JSEG=J
      IT=ISEG
      ISEG=ISEG+JSEG
      JT=JSEG
C**** ****************************************************************
C     MERGE EDGES OF FEATURE INTO LIST BY 'Y' VALUE
C**** ****************************************************************
C**** ****************************************************************
C     IF THIS IS THE FIRST EDGE OF THE FEATURE DUMP THE 'JX' ARRAY TO
C     THE 'IX' ARRAY
C**** ****************************************************************
  603 IF(IT.EQ.0)NJ=4*JT
      IF(IT.EQ.0)GO TO 605
C**** ****************************************************************
C     NI THE TOP OF THE IX LIST
C     NJ THE TOP OF THE 'JX' LIST
C**** ****************************************************************
C**** ****************************************************************
C     WHICH ARRAY (IX OR JX) HAS THE LARGEST Y VALUE
C**** ****************************************************************
      NI=4*IT
      NJ=4*JT
      IF(IX(NI+3)-JX(NJ-3))605,601,602
C**** ****************************************************************
C     IF Y'S ARE EQUAL CHECK FOR LARGEST X VALUE
C**** ****************************************************************
  601 IF(IX(NI+4)-JX(NJ-2))605,604,602
C**** ****************************************************************
C     IF X'S AND Y'S ARE EQUAL CHECK FOR LARGEST DELTA X VALUE
C**** ****************************************************************
  604 IF(IX(NI+5)-JX(NJ-1))605,625,602
C**** ****************************************************************
C     IF Y,X,DELTA X'S ARE EQUAL CHECK FOR LARGEST THETA VALUE
C**** ****************************************************************
  625 IF(IX(NI+6)-JX(NJ).GT.0)GO TO 602
C**** ****************************************************************
C     IIT THE TOTAL NUMBER OF SEGMENTS IN BOTH THE JX + IX BUFFERS
C     NII THE TOTAL NUMBER OF ENTRIES IN THE JX + IX BUFFERS
C**** ****************************************************************
C**** ****************************************************************
C     FOR THE LARGER VALUE IN THE JX ARRAY TRANSFER LAST ELEMENT OF
C     THE JX ARRAY TO THE END OF IX ARRAY THEN LOOP BACK TO GET ALL
C     ENTRIES IN THE JX ARRAY
C**** ****************************************************************
  605 IIT=IT+JT
      NII=4*IIT
```

```
      DO 598 K=1,4
      IX(NII+K+2)=JY(NJ+K-4)
  598 CONTINUE
      JT=JT-1
      IF(JT-1)250,603,603
C**** ****************************************************************
C     FOR LARGER VALUE IN THE IX ARRAY MOVE OLD IX ENTRY TO THE
C     TAILEND OF THE IX ARRAY THEN LOOP BACK TO GET NEXT ENTRY
C**** ****************************************************************
  602 IIT=IT+JT
      NII=4*IIT
      DO 599 K=1,4
      IX(NII+K+2)=IX(NI+K+2)
C**** ****************************************************************
C     PROCESSING STOPS WHEN THE JX ARRAY IS EMPTY
C**** ****************************************************************
  599 CONTINUE
      IT=IT-1
      GO TO 603
  250 CONTINUE
      IX(4*ISEG+7)=0
      IFILL=ISEG
      I=1
      J=1
      K=1
C**** ****************************************************************
C     FILL POLYGONS
C**** ****************************************************************
  251 CONTINUE
C**** ****************************************************************
C     IS 1ST EDGE A LEFT EDGE
C     IX(10) IS FIRST THEATA VALUE
C**** ****************************************************************
      IF(IX(10).GE.(-IRES)) GO TO 301
      IFG=-1
      GO TO 303
C**** ****************************************************************
C     IS 1ST EDGE A NEUTRAL (HORIZONTAL)
C**** ****************************************************************
  301 IF(IX(10).GE.(IRES)) GO TO 302
      IFG=0
      GO TO 303
C**** ****************************************************************
C     IF IT IS A POLYGON AN ERROR EXISTS SINCE CW + SORT REQUIRES
C     THE FIRST EDGE TO BE NON RIGHT
C**** ****************************************************************
  302 IF(ITYP.EQ.3) NI=4
      IF(ITYP.EQ.3)GO TO 327
C**** ****************************************************************
C     ARE THE Y'S EQUAL
C**** ****************************************************************
  303 NI=4*I
      NIJ=4*(I+J)
```

```
      IF(IX(NI+3).NE.IX(NIJ+3))GO TO 304
C**** ****************************************************************
C     AS A RESULT OF THE BOUNDRY ALGORITHM CORNERS MAY OVERLAP OR HAVE
C     GAPS
C     DOES X(I) + DX(I) GE X(I+1)
C**** ****************************************************************
      IF(IX(NI+4)+IX(NI+5).GE.IX(NIJ+4))GO TO 307
C**** ****************************************************************
C     THERE IS A GAP - IS IT A POLYGON + LEFT EDGE
C**** ****************************************************************
      IF(ITYP.EQ.3.AND.IFG.LT.0) GO TO 310
C**** ****************************************************************
C     IS FILL REQUIRED
C**** ****************************************************************
  308 IF(K.GE.I)GO TO 323
      NK=4*K
      DO 324 II=3,6
      IX(NK+II)=IX(NI+II)
  324 CONTINUE
  323 CONTINUE
      K=K+1
      IF(IX(NIJ+3).GE.(-IRES)) GO TO 345
      IFG=IFG-1
      GO TO 346
  345 IF(IX(NIJ+3).GE.IRES) IFG=IFG+1
  346 I=I+J
      J=1
C**** ****************************************************************
C     HAVE ALL SEGMENTS OF THE FEATURE BEEN HANDLED
C**** ****************************************************************
      IF(I.LE.ISEG)GO TO 303
  325 IREM=ISEG
      ISEG=K-1
      GO TO 500
C**** ****************************************************************
C     NOT A POLYGON OR IT IS HORIZONTAL
C**** ****************************************************************
  304 IF(ITYP.NE.3.OR.IFG.EQ.0)GO TO 308
C**** ****************************************************************
C     ERROR CONDITION LISTING
C**** ****************************************************************
  327 PRINT 328, I,ISEQ,IF,ISEG,ISPEC,ITYP,NI,(IX(NI+N),N=3,6)
  328 FORMAT(' R_MS2 ERROR ', 11I5)
      GO TO 325
C**** ****************************************************************
C     IF SPECULAR AND THETA (I) NE THETA(I+1)
C**** ****************************************************************
  307 IF((ISPEC.EQ.1).AND.(IX(NI+6).NE.IX(NIJ+6))) GO TO 313
  315 IX(NI+5)=IX(NIJ+5)+IX(NIJ+4)-IX(NI+4)
  316 IF(IX(NIJ+3).GE.(-IRES)) GO TO 317
      IFG=IFG-1
      GO TO 318
  317 IF(IX(NIJ+3).GE.IRES)IFG=IFG+1
```

```
  318 J=J+1
      GO TO 303
  310 IF(ISPEC.EQ.2)GO TO 315
C**** *****************************************************************
C     NOT SPECULAR - CALCULATE LENGTH OF FILLER USE X(I) FOR X
C**** *****************************************************************
  312 IFILL=IFILL+1
      NF=4*IFILL
      IX(NF+3)=IX(NI+3)
      IX(NF+6)=0
      IX(NF+4)=IX(NI+4)+IX(NI+5)
      IX(NF+5)=IX(NIJ+4)-IX(NF+4)
      GO TO 308
C**** *****************************************************************
C     IF X(I) + DX(I) GT X(I+1) GO TO 320
C**** *****************************************************************
  313 IF(IX(NI+4)+IX(NI+5).GT.IX(NIJ+4))GO TO 320
  322 IF(IX(NIJ+5).GT.0)GO TO 309
      GO TO 315
C**** *****************************************************************
C     INCREASE X(I+1) BY 1 + DECREASE DX(I+1) BY 1
C**** *****************************************************************
  320 IX(NIJ+4)=IX(NIJ+4)+1
      IX(NIJ+5)=IX(NIJ+5)-1
      GO TO 313
  330 PRINT 341
  341 FORMAT(' INPUT ERROR-RESOLUTION ELEMENT TOO FINE')
      STOP
  332 PRINT 333
  333 FORMAT(' INPUT ERROR-REGION SIZE TOO LARGE')
      STOP
  987 WORD(1)=IRGSIZ
      WORD(2)=IRGSZ
      BUFFER OUT(IN2W,1) (WORD(1),WORD(2))
      IF(UNIT(IN2W))986,985,985
  985 PRINT 986, IN2W
  986 FORMAT(' OUTPUT ERROR ON UNIT  ',I2)
  340 FORMAT('IN FEATURE NO.',I4,' ERROR AT Y=',I4)
C 986 PRINT 16, IIRC,ISEQ,ISET
C  16 FORMAT(' IIRC=', I4, '  ISEQ=', I4, '  ISET=', I4/
C 161        '...AT END OF PROGRAM.')
  988 PRINT 16
   16 FORMAT(//70X,'*****  SUCCESSFUL END OF RSS PROGRAM 2  *****')
      ENDFILE DISC
      REWIND DISC
      END
```

```
      PROGRAM RLMS3(OUTPUT,TAPE5,TAPE6,TAPE1,TAPE7=0,TAPE20=0)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C                PROGRAM RLMS3: DIVIDE FEATURES INTO REGIONS
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C                         *****************
C                         *****************
C     THIS PROGRAM DIVIDIES THE FEATURES DEFINED BY Y-X-DELTA X-THETA
C     SEGMENTS INTO REGIONS
C                         *****************
C                         *****************
      COMMON IN(15000)
      DIMENSION JY(500),JX(500),JDX(500),JTH(500),IFC(33)
      INTEGER WORD(10)
      INTEGER ROTTH
      INTEGER OUTAP,DISC
      DATA OUTAP/1/,DISC/7/
      DATA IN2W/20/
C                         *****************
C     TRANSLATION TABLE FEATURE CODE TO MERGE PRIORITY
C     FEATURE NUMBER 13 IS TREATED LIKE FEATURE NUMBER 9
C                         *****************
      DATA IFC/2.,19,24,7,11,20,7,21,16,11,22,1,8,10,18,14,7,17,15,4,
     *9,3,8,12,17,13,3,1,2,6,5,5,23/
C                         *****************
C     NX ELEMENTS PER X DIMENSION OF REGION
C     NY ELEMENTS PER Y DIMENSION OF REGION
C     NRX THE NUMBER OF REGIONS IN THE X DIMENSION OF MAP
C                         *****************
      BUFFER IN(IN2W,1) (WORD(1),WORD(2))
      IF(UNIT(IN2W)) 102,985,985
  102 NY=NX=WORD(2)
      NRX=331
      INRC=INSG=0
      IRES=2048
C                         *****************
C     READ IN FEATURE
    1 IST=1
      MWD=0
  103 BUFFER IN (DISC,1)  (IN(IST),IN(IST+1799))
      INDIC=UNIT(DISC)
      NWD=LENGTH(DISC)
      IF(INDIC) 3,44,987
    3 MWD=MWD+NWD
      IF(MWD - (4*IN(4)+6)) 4,6,5
    4 IST=IST+NWD
      GO TO 103
C                         *****************
C     CHECK NUMBER OF WORDS TRANSFERED
C                         *****************
    5 PRINT 10, MWD,(IN(J),J=1,10)
   10 FORMAT(I5, ' WORDS INPUT, FIRST TEN ARE ', 10I5)
```

```
              GO TO 1
      C                      *****************
      C       REGENERATE CODES FROM FLAGS
      C                      *****************
            6 ISEQ=IN(1)$TF=IN(2)$ISLG=IN(4)$ISPEC=IN(5)$ITYP=IN(6)
              ILIM=4*ISEQ+6
              DO 33 I=10,ILIM,4
      C                      *****************
      C       MODIFY THETA TO ELIMINATE FLAG
      C       THETA IS REPRESENTED BY -2048 TO +2047
      C                      *****************
              IF(IN(I).GT.IRES)IN(I)=IN(I)-2*IRES
              IF(IN(I).LT.(-IRES))IN(I)=IN(I)+2*IRES
           33 CONTINUE
      C                      *****************
      C
      C       FIND THE REGION BOUNDS FOR THE TOP AND BOTTOM OF THE FEATURE
      C       VIA THE Y SORT OF THE DATA
      C                      *****************
              ILIM=4*ISEQ+3
              IYL1=(IN(7)-1)/NY+1
              IYL2=(IN(ILIM)-1)/NY+1
              DO 33 III=IYL1,IYL2
      C                      *****************
      C       FIND THE MAX AND MIN X VALUE FOR EACH Y BAND
      C                      *****************
              II=NY*III
              II=NY*(III-1)
              IMIN=IMAX=IN(8)
                 DO 13 I=7,ILIM,4
      C          IF((IN(I).LT.II).OR.(IN(I).GE.II+NY))GO TO 13
              IF((IN(I).LE.II).OR.(IN(I).GT.II+NY)) GO TO 13
      C          IF(IMAX.LT.IN(I+1)+IN(I+2))IMAX=IN(I+1)+IN(I+2)
              IF(IMAX.LE.IN(I+1)+IN(I+2)-1)  IMAX=IN(I+1)+IN(I+2)-1
                 IF(IN(I+1).LT.IMIN)IMIN=IN(I+1)
           13    CONTINUE
      C                      *****************
      C       FIND REGION NUMBER FOR MIN AND MAX X'S
      C       DEFINE THE X + DELTA X SEGMENTS FOR EACH REGION IN BAND
      C       II IS THE Y VALUE AND JJ IS THE X VALUE FOR THE CORNER OR THE
      C       REGION
      C                      *****************
              IXL1=(IMIN-1)/NX+1
              IXL2=(IMAX-1)/NX+1
                 DO 42 JJJ=IXL1,IXL2
                 JJ=NX*JJJ
              JJ=NX*(JJJ-1)
                 J=0
                     DO 18 I=7,ILIM,4
      C                      *****************
      C       IF DATA IS NOT IN REGION GO TO NEXT DATA ITEM
      C                      *****************
      C              IF((IN(I).LT.II).OR.(IN(I).GE.II+NY).OR.(IN(I+1).LT.JJ).OR.
      C     1        (IN(I+1).GE.JJ+NX))GO TO 18
```

```
      IF((IN(I).LE.II).OR.(IN(I).GT.II+NY).OR.(IN(I+1).LE.JJ).OR.
     1(IN(I+1).GT.JJ+NX)) GO TO 18
C                    *****************
C     CONVERT FROM MAP COORDINATES TO REGION COORDINATES
C     J IS THE NUMBER OF SEGMENTS PLACED IN THE (J) ARRAY'S
C                    *****************
            J=J+1$JY(J)=IN(I)-II$JX(J)=IN(I+1)-JJ$JTH(J)=IN(I+3)
C                    *****************
C     DELTA X IS SAME IF TOTALLY WITHIN REGION
C                    *****************
C           IF((IN(I+1)+IN(I+2)).GT.JJ+NX) GO TO 17
      IF((IN(I+1)+IN(I+2)-1).GT.JJ+NX) GO TO 17
      JDX(J)=IN(I+2)
      GO TO 18
C                    *****************
C     COMPUTE DELTA X WITHIN REGION, DECREMENT REMAINING DELTA X AND SET
C     VALUE OF SEGMENT TO THE VALUE OF THE REGION BOUNDRY
C                    *****************
C 17        JDX(J)=NX-JX(J)$IN(I+2)=IN(I+2)-JDX(J)$IN(I+1)=JJ+NX
   17 JDX(J)=NX-JX(J)+1$IN(I+2)=IN(I+2)-JDX(J)$IN(I+1)=JJ+NX+1
C                    *****************
C     IF NO SEGMENTS SKIP OUT
C                    *****************
   18       CONTINUE
        IF(J.EQ.0) GO TO 42
C                    *****************
C     FIND REGION NUMBER AND COUNT SEGMENTS
C     III IS THE REGION COUNT IN THE Y DIRECTION
C     JJJ IS THE REGION COUNT IN THE X DIRECTION
C                    *****************
C THIS WILL HOPEFULLY CORRECT SHIFT OF CULTURE DATA
C        IREG=NRX*III+JJJ+1$INSG=INSG+J$K=0
      IREG=NRX*(III-1)+JJJ$INSG=INSG+J$K=0
C                    *****************
C     PLACE SEVEN SEGMENTS ON CARD IMAGES
C                    *****************
   31   NUM=J$IF(J.GT.7)NUM=7
C                    *****************
C     WRITE DATA TO TAPE AS CARD IMAGES
C                    *****************
      WRITE(OUTAP,30) IREG,IFC(IF),IF,ITYP,ISPEC,NUM,
     1JY(K+1),JX(K+1),JDX(K+1),JTH(K+1),
     2JY(K+2),JX(K+2),JDX(K+2),JTH(K+2),
     3JY(K+3),JX(K+3),JDX(K+3),JTH(K+3),
     4JY(K+4),JX(K+4),JDX(K+4),JTH(K+4),
     5JY(K+5),JX(K+5),JDX(K+5),JTH(K+5),
     6JY(K+6),JX(K+6),JDX(K+6),JTH(K+6),
     7JY(K+7),JX(K+7),JDX(K+7),JTH(K+7)
   30 FORMAT(I5,2I2,3I1,7(3I2,I5))
C                    *****************
C     INCREMENT K SUMMARIZE SEGMENTS AND RECORDS USED
C                    *****************
        K=K+7$INRC=INRC+1$J=J-7$IF(J.GT.0)GO TO 31
```

```
   42   CONTINUE
C               *****************
C       GET THE NEXT FEATURE
C               *****************
   39 CONTINUE
      GO TO 1
C               *****************
C       JOB COMPLETE PRINT SUMMARIES
C               *****************
   44 PRINT 46, INRC,INSG
   46 FORMAT(1H0,' SUMMARY ',I6,'  RECORDS,',I6,'  SEGMENTS')
      ENDFILE OUTAP
      REWIND OUTAP
      REWIND DIST
      PRINT 7702
 7702 FORMAT(1H0,'    SUCCESSFUL END OF PLMS3')
      STOP
C               *****************
C       BUFFER IN ERROR LIST
C               *****************
  987 PRINT 50, ISEQ
   50 FORMAT(' BUFFER IN ERROR AT ISEQ=', I7)
      GO TO 44
  985 PRINT 986, IN2W
  986 FORMAT(' INPUT ERROR ON UNIT ',I2)
      END
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

SORTMRG,      RSS4
SORT,VAR=DISK
BYTESIZE,6
FILE,SORT=TAPE1,OUTPUT=TAPE2
FIELD,REGION(1,1,5,DISPLAY),PRIORITY(6,1,2,DISPLAY),
,INTY(13,1,2,DISPLAY),INTA(15,1,2,DISPLAY)
KEY,REGION(A,OWN),PRIORITY(A,OWN),INTY(A,OWN),INTY(A,OWN)
SEQUENCE,OWN( ,0,1,2,3,4,5,6,7,8,9)
OPTIONS,RETAIN
END
```

COPY AVAILABLE TO DDC DOES NOT PERMIT FULLY LEGIBLE PRODUCTION

```
      PROGRAM RLMS5 (INPUT,OUTPUT,TAPE6=OUTPUT,TAPE7,TAPE10,TAPE11,
     .TAPE60)
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C        PROGRAM RLMS5: MERGE FEATURES
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      COMMON IX(500),IY(500),IZ(500),ITH(500),IF(500),IG(500)
     1,       JX(500),JY(500),JZ(500),JTH(500),JF(500),JG(500)
     2,       KX(500),KY(500),KZ(500),KTH(500),KF(500),KG(500)
     3,       INDEX(28467),JREG
      INTEGER WORD(10)
      INTEGER DISC $DATA DISC/7/
      DATA IN2W/20/
C     ***************************************************************
C     OPEN THE FILE FOR SUBROUTINE RADOUT
C     ***************************************************************
         CALL OPENMS(10,INDEX,28467,0)
      NY=96$NX=351
      NEMT=0
      NFUL=0
      NTOT=0
      NSEG=0
      NBYT=0
      IEND=0
      JN=0
      JREG=0
      BUFFER IN(IN2W,1) (WORD(1),WORD(2))
      IF(UNIT(IN2W)) 10,90,90
   10 IKC=WORD(2)
      IKC2=WORD(2)**2
    1 READ(DISC,2) IREG,MERGE,IF(1),IGG,IKK,IN
     *             ,((IX(I),IY(I),IZ(I),ITH(I)),I=1,7)
    2 FORMAT(I5,2I2,3I1,7(3I2,I5))
      IF(EOF(DISC) .NE.0) GO TO 31
      DO 4 I=1,IN
      IF(I)=IF(1)
      IG(I)=IKK
    4 CONTINUE
      IF(IREG-JREG)68,6,5
   68 PRINT 69, IREG,JREG
   69 FORMAT(3X,'REGION',I6,' FOLLOWS REGION',I6)
    5 IF(JN)8,7,8
C     ***************************************************************
C     ALL THE SEGMENTS FOR A REGION HAVE BEEN MERGED ... WRITE THEM OUT
C     ***************************************************************
    8 K=JN
      CALL RADOUT(K)
      NFUL=NFUL+1
      NBYT=NBYT+K
      IF(IEND) 7,7,30
C****************************************************************
C     ARRAYS BEGINNING IN J WILL CONTAIN THE INFORMATION FOR THE SEGMENTS IN
C     THE REGION BEING PROCESSED. INITIALIZE THESE SO THAT REGION IS REPRESENTED
C     BY 32 SEGMENTS WHICH FILL THE ENTIRE AREA. VALUES FOR INITIALIZATION ARE
```

COPY AVAILABLE TO DDC DOES NOT PERMIT FULLY LEGIBLE PRODUCTION

```
C     CHOSEN SO THAT LATER LOGIC WILL ALWAYS OVERWRITE THESE INITIAL SEGMENTS
C     WHEN OVERLAPPING CULTURE DATA EXISTS
C**********************************************************************
    7 DO 11 J=1,IKO
      JX(J)=J
      JY(J)=1
C     JY(J)=0
      JZ(J)=IKO
      JTH(J)=0
      JF(J)=35
      JG(J)=1
   11 CONTINUE
      JN=IKO
   33 JREG=JREG+1
C**********************************************************************
C     IF REGION CONTAINS NO CULTURE DATA, OUTPUT AN EMPTY RECORD
C     LATER LOGIC TO MERGE ELEVATION AND CULTURE DATA WILL REQUIRE SUCH A
C     FILLER RECORD
C**********************************************************************
      IF(IREG-JREG)34,6,34
   34 K=0
      CALL RADOUT(K)
      NEMT=NEMT+1
      GO TO 33
    6 I=1
      J=1
      K=0
C**********************************************************************
C     ARE THE Y-VALUES FOR THE SEGMENTS EQUAL
C     IF YES, FURTHER PROCESSING IS REQUIRED
C     IF NO, SEGMENT WITH SMALLER Y-VALUE GOES INTO K-ARRAYS
C**********************************************************************
   15 IF(IX(I)-JX(J))12,13,14
C**********************************************************************
C     IF THE ENDPOINT OF THE SEGMENT IN THE J-ARRAYS IS SMALLER THAN OR EQUAL
C     TO THE BEGINNING OF THE SEGMENT IN THE I-ARRAY, NO OVERLAP CAN OCCUR.
C     FILL THE K-ARRAYS FROM THE J-ARRAYS
C**********************************************************************
   13 IF(JY(J)+JZ(J)-IY(I))14,14,16
C**********************************************************************
C     IF ENTIRE SEGMENT IN I-ARRAYS LIES BEFORE THE BEGINNING POINT OF THE
C     SEGMENT IN THE J-ARRAYS, FILL K-ARRAYS FORM I-ARRAYS. NOTE THAT SEGMENTS
C     IN K-ARRAYS ARE ORDERED BY INCREASING Y AND X.
C**********************************************************************
   16 IF(IY(I)+IZ(I)-JY(J))12,12,17
C**********************************************************************
C     IF THERE IS OVERLAP, WITH THE IARRARY SEGMENT BEGINNING INSIDE THE JARRAY
C     SEGMENT, FILL THE K-ARRAYS WITH THE PORTION OF THE JARRAY SEGMENT TO THE
C     LEFT OF THE BEGINNING OF THE IARRAY SEGMENT
C**********************************************************************
   17 IF(IY(I)-JY(J))18,18,19
   18 IF(JY(J)+JZ(J)-IY(I)-IZ(I))20,20,21
   20 J=J+1
```

```
   47 IF(J-JN)48,48,49
   48 IF(I-IN)15,15,14
   49 IF(I-IN)12,12,23
C*************************************************************************
C     ARRAYS PREFIXED BY K FORM A WORK AREA WHERE THE J-ARRAYS (CONTAINING THE
C     EXISTING REGION SEGMENTS) ARE MERGED WITH THE I-ARRAYS (CONTAINING THE
C     NEWEST SEGMENT).
C*************************************************************************
   12 K=K+1
      KX(K)=IX(I)
      KY(K)=IY(I)
      KZ(K)=IZ(I)
      KTH(K) = ITH(I)
      KF(K)=IF(I)
      KG(K)=IG(I)
   53 I=I+1
      IF(I-IN)50,50,47
   50 IF(IX(I)-IX(I-1))47,51,47
   51 IF(IY(I)-IY(I-1))47,52,47
   52 IF(IZ(I)-IZ(I-1))47,53,47
   21 JZ(J)=JY(J)+JZ(J)-IY(I)-IZ(I)
      JY(J)=IY(I)+IZ(I)
      GO TO 12
C     ************************************************************************
C     OUTPUT TO THE K-ARRAYS THE PORTION OF THE J-ARRAY SEGMENT TO THE LEFT
C     OF THE OVERLAPPING I-ARRAY SEGMENT
C     DEFINE A NEW J-ARRAY SEGMENT, SHORTENED BY THE LENGTH OF THE OUTPUT PART
C     ************************************************************************
   19 K=K+1
      KX(K)=JX(J)
      KY(K)=JY(J)
      KZ(K)=IY(I)-JY(J)
      KTH(K)=JTH(J)
      KF(K)=JF(J)
      KG(K)=JG(J)
      JZ(J)=JZ(J)+JY(J)-IY(I)
      JY(J)=IY(I)
      GO TO 15
C     ************************************************************************
C     FILL THE K-ARRAYS FROM THE J-ARRAYS
C     ************************************************************************
   14 K=K+1
      KX(K)=JX(J)
      KY(K)=JY(J)
      KZ(K)=JZ(J)
      KTH(K)=JTH(J)
      KF(K)=JF(J)
      KG(K)=JG(J)
      GO TO 20
C     ************************************************************************
C     THE PRESENT GROUP OF 7 SEGMENTS HAVE BEEN MERGED
C     COPY THE WORK AREA INTO THE REGION ARRAYS, THUS DEFINING A NEW INITIAL
C     REGION STRUCTURE AND RETURN FOR MORE INPUT DATA
```

```
      C    ************************************************************************
        23 ISUM=0
           DO 25 N=1,K
           JX(N)=KX(N)
           JY(N)=KY(N)
           JZ(N)=KZ(N)
           JTH(N)=KTH(N)
           JF(N)=KF(N)
           JG(N)=KG(N)
           ISUM=ISUM+JZ(N)
        25 CONTINUE
           JN=K
           IF(ISUM-IN32) 26,1,26
        26 PRINT 23,     IREG,ISUM,I-1,J-1,K
        23 FORMAT(3X,'IN REGION',I6,', SUM='I5,', MERGE',I4,'+',
          1I4,' TO GIVE',I4)
           GO TO 1
        31 IEND=1
           GO TO 8
        30 CONTINUE
           IREG=NY*NY
        83 JREG=JREG+1
           IF(IREG-JREG) 84,86,84
        84 K=0
           CALL RADOUT(K)
           NEMT=NEMT+1
           GO TO 83
        86 CONTINUE
           CALL CLOSMS(10)
           BUFFER OUT(11,1)(INDEX(1),INDEX(28467))
           IF(UNIT(11)) 87,87,88
        88 WRITE(6,89)
        89 FORMAT(' PARITY ERROR DURING INDEX WRITE-OUT')
        87 CONTINUE
           PRINT 9
         9 FORMAT('/END PLMS5 ... CULTURE PREPARATION COMPLETE')
           STOP
        90 PRINT 91, IN2W
        91 FORMAT(' INPUT ERROR ON UNIT ',I2)
           END
```

```
      SUBROUTINE RADOUT(K)
      COMMON IX(500),IY(500),IZ(500),ITH(500),IF(500),IG(500)
     1,       JX(500),JY(500),JZ(500),JTH(500),JF(500),JG(500)
     2,       KX(500),KY(500),KZ(500),KTH(500),KF(500),KG(500)
     3,       INDEX(28667),JREG
      INTEGER DATA(500)
      JK=1
      DATA(1)=K
      IF(K.EQ.0)GO TO 2
      DO 1 I=1,K
      IF(JF(I).EQ.35) GO TO 1
      JK=JK+1
      IF(JG(I).EQ.0) JTH(I)=2047
C
C        IN XEROX SDS VERSION, EACH WORD OF -DATA- WAS 32 BITS;
C        VALUES WERE PACKED WITH 4 AT 5 BITS AND 1 AT 12 BITS (20+12=32)
C        WITH THE FOLLOWING STATEMENT....
C
C     DATA(JK)=IOR(ISL(JX(I),27),ISL(JY(I),22),ISL(JZ(I)-1,17),
C    1      (ISL(IAND(JTH(I),.Z00000FFF),5)),JF(I))
C
C        FOR CDC FORTRAN, THE SAME PACKING INTO 32 BITS IS DONE BUT USING
C     DIFFERENT FORTRAN FEATURES AS FOLLOWS (RESULT IS A 60-BIT WORD
C     OF COURSE).....
      DATA(JK)=(JX(I)*10000000000B) .OR.
     +         (JY(I)*100000000B) .OR.
     +         (JZ(I)*1000000B) .OR.
     *         ((JTH(I).AND.7777B)*100B).OR.JF(I)
    1 CONTINUE
      DATA(1)=JK-1
    2 CONTINUE
      CALL WRITMS(10,DATA,JK,JREG)
      RETURN
      END
```

COPY AVAILABLE TO DDC DOES NOT PERMIT FULLY LEGIBLE PRODUCTION

```
      PROGRAM RLM36  (INPUT,OUTPUT,TAPE6=OUTPUT,TAPE7,TAPE8=0,TAPE9=0,
     .TAPE20=0)
      DIMENSION INDEX(8301),IARRAY(12,48),INBUF(800,48)
      DIMENSION MULT(2),MOVE15(4)
      INTEGER WORD(10)
      INTEGER XCOOR,YCOOR,XFILL,YFILL,AJUP,OUT( 577),OUTAP,OUTPUT
      DATA INTAP/8/,OUTAP/9/,IMASK/77777B/
      DATA MULT/1,1000000000B/,MOVE15/45,30,15,0/
      DATA IN2W/20/
C***********************************************************************
C     READ HEADER FROM INPUT TAPE
C***********************************************************************
      BUFFER IN (IN2W,1) (WORD(1),WORD(2))
      IF(UNIT(IN2W)) 14,500,777
   14 CONTINUE
   15 BUFFER IN(INTAP,1)(INBUF(1,1),INBUF(36,1))
      IF(UNIT(INTAP)) 1,500,777
C***********************************************************************
C     CALCULATE MAP-SIZE PARAMETERS FROM HEADER INFORMATION
C***********************************************************************
    1 IF(INBUF(34,1).NE.WORD(1)) GO TO 200
      IK=WORD(2)
      IL = IK/2
      IM=IK**2/4+1
      IN = IK/4
      NROW=INBUF(21,1)/1000 $NCOL=INBUF(22,1)/1000
      YFILL=MOD(NROW,IK) $XFILL=MOD(NCOL,IK)
      NYREG=(NROW/IK)+1 $IF(YFILL.EQ.0) NYREG=NYREG-1
      NXREG=(NCOL/IK)+1 $IF(XFILL.EQ.0) NXREG=NXREG-1
      YCOOR=NYREG $XCOOR=1 $NREC=YFILL
      IF(YFILL.EQ.0) NREC=IK
      PRINT 70, XCOOR,YCOOR,NREC,NXREG,NYREG
   70 FORMAT('0 INITIAL XCOOR YCOOR NREC NXREG NYREG',5I12)
      KOUNT=0
C***********************************************************************
C     WRITE HEADER TO OUTPUT TAPE
C     OPEN DISK FILE TO BE USED FOR TEMPORARY STAORAGE OF REGION RECORDS
C***********************************************************************
      BUFFER OUT(OUTAP,1) (INBUF(1,1),INBUF(36,1))
      IF(UNIT(OUTAP)) 6,500,777
    6 CALL OPENMS(7,INDEX,8301)
C***********************************************************************
C     READ ELEVATION DATA FOR ONE STRIP OF REGIONS
C***********************************************************************
    3 DO 2 II=1,NREC
      JJ=NREC-II+1
      BUFFER IN(INTAP,1)(INBUF(1,JJ),INBUF(800,JJ))
      IF(UNIT(INTAP)) 2,500,777
    2 CONTINUE
C***********************************************************************
C     TRANSFER POINTS FROM INBUF(I,J) TO ARRAY(I,J)
C     ARRAY WILL CONTAIN THE 24X24 POINTS NEEDED TO DESCRIBE ONE REGION
C     DATA IN ARRAY IS PACKED TWO ELEVATIONS PER WORD ... DATA IN INBUF
```

```
C     IS PACKED FOUR ELEVATIONS PER WORD
C     NREC IS MADE VARIABLE IN ORDER TO IMPLIMENT A BORDER ON THE NORTHERN
C     EDGE OF THE MAP. A SIMILAR BORDER WILL ALSO APPEAR AUTOMATICALLY
C     ON THE EASTERN EDGE. THIS IS NECESSARY SO THAT THE MAP WILL
C     CONTAIN AN INTEGER NUMBER OF REGIONS
C*********************************************************************
      PRINT 72, XCOOR,YCOOR
   72 FORMAT(*  STARTED NEW ROW OF REGIONS WITH XCOOR AND YCOOR*,2I15)
    5 DO 10 J=1,NREC
      IPOINT=1+IN*(XCOOR-1)
      DO 10 I=1,IN
      IARRAY(I,J)=INBUF(IPOINT,J)
      IPOINT=IPOINT+1
   10 CONTINUE
C*********************************************************************
C     CALCULATE THE REGION NUMBER ASSOCIATED WITH EACH REGION BASED ON
C     A MAXIMUM SIZE OF 331 REGIONS IN THE X DIRECTION
C*********************************************************************
      NREGION=331*(YCOOR-1)+XCOOR
      OUT(1)=NREGION
      NREGION=100*(YCOOR-1)+XCOOR
C*********************************************************************
C     TRANSFER INFORMATION FROM ARRAY(I,J) TO OUT(I)
C     EACH STRING OF 12 WORDS IN OUT CONTAINS THE INFORMATION FOR ONE ROW
C     OF THE REGION ... BEGINNING AT THE LOWER LEFT
C*********************************************************************
      DO 30 I=1,IN
      DO 30 J=1,IK
      K=IN*(J-1)+I+1
   30 OUT(K)=IARRAY(I,J)
C*********************************************************************
C     WRITE REGION INFORMATION OUT TO DISK
C*********************************************************************
      CALL WRITMS(7,OUT,IM,NREGION)
      IF(YCOOR.EQ.1.AND.XCOOR.EQ.NXREG) GO TO 100
      IF(XCOOR.NE.NXREG) GO TO 40
      NREC=IK $XCOOR=1 $YCOOR=YCOOR-1 $GO TO 3
   40 XCOOR=XCOOR+1
      GO TO 5
C*********************************************************************
C     ALL REGIONS HAVE BEEN WRITTEN TO DISK
C     READ THEM BACK IN IN REGION ORDER AND WRITE TO TAPE
C*********************************************************************
  100 KOUNT1=0
      DO 150 J=1,NYREG
C 100 DO 150 J=1,NYREG
      DO 150 I=1,NXREG
      ADDR=100*(J-1)+I
      CALL READMS(7,OUT,IM,ADDR)
      BUFFER OUT(OUTAP,1)(OUT(1),OUT(IM))
      IF(UNIT(OUTAP))1150,777,777
 1150 CONTINUE
      KOUNT1=KOUNT1+1
```

```
      IF(MOD(KOUNT1,10)) 153,151,153
  151 PRINT 152, (OUT(IJK),IJK=1,5),J,I
  152 FORMAT(' OUT',5O20,2I10)
  153 CONTINUE
  150 CONTINUE
      PRINT 350
  350 FORMAT(' RLMSF SUCESSFUL END')
      STOP
  500 WRITE(6,501)
  501 FORMAT(' PREMATURE EOF ENCOUNTERED ON INPUT TAPE')
      STOP
  777 WRITE(6,778)
  778 FORMAT(' PARITY ERROR ENCOUNTERED ... ERROR EXIT')
      STOP
  200 PRINT 510
  510 FORMAT(' ERROR-CULTURE RESOLUTION NOT EQUAL TO TERRAIN RESOL.')
      END
```

```
      PROGRAM RLMS7(INPUT,OUTPUT,TAPE1,TAPE7,TAPE12,TAPE6=0,TAPE20=0)
C
C " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
C         PROGRAM RLMS7 = LTL6  MERGE CULTURE AND TERRAIN DATA FOR
C                                  GIVEN RADAR PARAMTERS
C " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
C
C + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
C                  VARIABLE USAGE
C + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
C
C***** ARRAYS *****
      DIMENSION MOVE(4)
      DIMENSION LDATA(5500)
      DIMENSION MSIX(23467)
      INTEGER TABLE(1251),DATA(1000)
      DIMENSION PLINK(5)
      INTEGER WORD(10)
      INTEGER REC
      DATA MOVE/3,45,30,15/
      DATA IN2W/20/
C TABLE   - ELEVATION DATA RECORD BUFFER, 2 HTS PER WD, FILED BY REGION.
C DATA    - CULTURE DATA RECORD FOR ON  REGION, 0-500 STRIPS, ENCODED AND PACKED
C LDATA   - PRINT LINE CHECK VALUES FOR CULTURE REGIONS PROCESSED
C MSIX*   - INDEX DIRECTORIES USED BY MASS STORAGE I/O
C
C***** SCALARS *****
C ITERR   (SET PARAMETER) -SWITCH TO ALLOW BYPASS OF TERRAIN DATA INPUT
C NX      (SET PARAMETER) -NMBR OF REGIONS ACROSS ON CULTURE GRID
C NY      (SET PARAMETER) -NMBR OF REGIONS VERTICAL ON CULTURE GRID
C NR      (SET PARAMETER) -NMBR OF REGION SENCOMPASSING RADAR CIRCLE
C ALTITDE (SET PARAMETER) -RADAR HEIGHT IN FEET
C REGNM   - REGIONS PER NAUTICAL MILE
C DEPNGL  - RADAR DEPRESSION ANGLE FUNCTION = 1.5+CTN25
C X,Y     (REQUIRED PARAM) -RADAR LOCATION, IN N.M. FROM ORIGIN
C RANGE   - RADAR RADIUS OF COVERAGE IN N.M.
C IX,IY   - COMPUTED COL AND ROW OF LOWER CORNER REGION OF AREA
C II,JJ   - INDEX FOR ROW/COL OF REGIONS IN AREA
C REG,IREG- CURRENT REGION CODE (IDENTICAL VALUES)
C
C + + + + + + + + + + + + + + + + + + + + + + + + +  + + +  + + + + + + + +
C
      NAMELIST /PARAMS/ X,Y,IFREQ,ITERR,ICULT,ALTITDE
C     SET DATA PARAMETERS
      DEPNGL=3.6443
C
C
      ALTITDE=3200.0.
C
C
      ICULT=1
      ITERR=1
      NY=331
```


COPY AVAILABLE TO DDC DOES NOT PERMIT FULLY LEGIBLE PRODUCTION

```
      IFRED=2
      INDEX=1
      CALL OPENMS(12,MSIX  ,28467,0)
      CALL OPENMS(3,MSTX,28467,0)
C
      READ PARAMS
      PRINT 80, X,Y,IFREQ,ITERP,ICULT,ALTIDE
   80 FORMAT(/////*** INPUT VALUES ARE X =*,F12.4,5X,*Y =*,F12.4,5X,
     .  *IFREQ =*,I5,5X,*ITERP =*,I4,5X,*ICULT =*,I4,5X,*ALTIDE =*,
     .F9.0/)
      BUFFER IN(INEW,1) (WORD(1),WORD(2))
      IF(UNIT(INEW)) 45,20,20
   45 CONTINUE
C
      IF(ITERP.EQ.0) GO TO 20
C
C       FETCH THE TERRAIN HEADER RECORD AND PROCESS
C
      BUFFER IN(1,1) (DATA(1),DATA(36))
      IF(UNIT(1)) 22,5,21
   21 PRINT 60
C         SAVE APPROPRIATE VALUES
   22 YLAT=FLOAT(IABS(DATA(23)))/1000.+FLOAT(DATA(24))/60000.
     *                                  +FLOAT(DATA(25))/3600000.
      S=FLOAT(DATA(23))
      YLAT=SIGN(YLAT,S)
      XLON=FLOAT(IABS(DATA(27)))/1000.+FLOAT(DATA(28))/60000.
     *                                  +FLOAT(DATA(29))/3600000.
      S=FLOAT(DATA(27))
      XLON=SIGN(XLON,S)
      PRINT 23, XLON,YLAT
   23 FORMAT(*0TARGET COORDS, LONG=*, F8.3, *, LAT=*, F8.3,)
      DP=(19.1945275.11549/DATA(34)/304.8)/(32000/ALTIDE)
      IF(DP-1000) 46,46,43
   43 PRINT 44
   44 FORMAT(* WARNING-RESOLUTION ELEMENT TOO FINE FOR THIS ALTITUDE*)
   46 CONTINUE
C
C      X AND Y ARE INPUT RELATIVE TO THE NW CORNER OF THE MAP AND MUST
C    BE CONVERTED RELATIVE TO THE SW CORNER.  HENCE THE TOTAL DISTANCE
C    IN THE Y DIRECTION MUST BE ACQUIRED AND USED.
C      ALSO, THE SIGN OF X AND Y GIVE DIRECTION OF TARGET RELATIVE TO
C    THE NW CORNER;  POSITIVE VALUES ARE WEST AND NORTH RESPECTIVELY,
C    NEGATIVES EAST AND SOUTH.  SO FOR TARGET TO LIE WITHIN MAP COVERAGE
C    AREA, BOTH X AND Y MUST BE INPUT AS NEGATIVE VALUES.
C
C        DETERMINE TARGET LOCATION AND SCALE TO N.M. FROM NW CORNER
C        DISTANCE IS (NR. RECS-1)*METERS PER ELMT CONVERTED TO N.M.
      YDIST=(((FLOAT(DATA(21))/1000.)-1.)*(FLOAT(DATA(34))/1000.))/
     .    1852.00
C        CONVERT TO N.M., Y RELATIVE TO SW CORNER
      IF(DATA(34).NE.WORD(1)) GO TO 700
      REST=DATA(34)
```

COPY AVAILABLE TO DDC DOES NOT PERMIT FULLY LEGIBLE PRODUCTION

```
      RES = (DATA(74)/3.048)/100.
      IK=NJPD(2)
      IM=IK**2/4+1
      REGNM = 6076.11549/(IK*RES)
      X=X/1852.004 $ Y=Y/1852.004
C     Y=YDIST-(Y/1852.004)
      ITX=X*6076.11549/RES+1
      ITY=Y*6076.11549/RES+1
      IXCOR=MOD(ITX-1,IK)+1
      IYCOR=MOD(ITY-1,IK)+1
C     IIREG=NX*((ITY-1)/IK)+(ITX-1)/IK+1
      ITREG=NX*((ITY-1)/IK)+(ITX)/IK+1
C
   28 RANGE=DEPNGL*ALTITDE/6076.11549
      IF(X-RANGE.GE.0.AND.Y-RANGE.GE.0) GO TO 30
      PRINT 29,X,Y,RANGE
   29 FORMAT(' RANGE TOO GREAT FOR COVERAGE AT SPECIFIED TARGET POSITION
     +'/' X=',F10.4,' Y=',F10.4,' RANGE=',F10.4)
C
C        DETERMINE REGION POSITION IN LOWER CORNER OF AREA OF COVERAGE.
C
   30 IX=REGNM*(X-RANGE)+1.
      IY=REGNM*(Y-RANGE)+1.
      ILIM=REGNM*(Y+RANGE)+1.
      JLIM=REGNM*(X+RANGE)+1.
      NR=MAX0((JLIM-IX),(ILIM-IY))
C        COMPUTE LINEAR ADDRESS OF FIRST REGION AFFECTED.
      NBASER=IX+ NX*(IY-1)
      PRINT 31,ALTIDE, DEPNGL, RANGE, X, Y, NR, NBASER
   31 FORMAT('1RLAST DATA EXTRACT FOR THE FOLLOWING RADAR PARAMETERS:'/
  311     ' HEIGHT DEPRESSION ANGLE RADIUS OF COVERAGE          LOCATIO
  312N(X,Y)    '/3X'(FT)',7X'(DEGR)',12X'(N.M.)',16X'(N.M.)'/
  313        F9.1, 5XF5.2, 12XF7.3, 8X 2F10.4/
  314        ' (NMBR OF REGIONS OF COVERAGE=' I3,5X 'STARTING REGION NMB
  315R= ',I3,')'/)
C        EXTRACT ALL THE REGIONS IN A NR X NR SQUARE ABOUT THE X,Y CENTER
      DO 1 I=IY,ILIM
      DO 2 J=IX,JLIM
      REG=J+NX*(I-1)
      IREG=REG
C        INPUT THE CULTURE PORTION
      IF(ICULT.EQ.0) GO TO 33
    3 CALL READMS(3      ,DATA,500,IREG)
      GO TO 34
   33 DATA(1)=0
   34 K1=DATA(1)+2
      IF(K1.GT.500) PRINT 70,K1,REG
   70 FORMAT(O20,' CULTURE STRIPS IN REGION ',I5)
      K2=DATA(1)+IM
C        SEE IF CULTURE DATA FOR THIS REGION EXISTS
      IF(DATA(1).LE.0) GO TO 41
C        FILL PRINT LINE VALUES
      LDATA(INDEX)=REG$LDATA(INDEX+1)=DATA(1)$INDEX=INDEX+2
```

```
C         BYPASS ELEVATION DATA IF TERRAIN SWITCH IS NOT ON
   41 IF(ITERR.EQ.1) GO TO 40
      DO 39 K=K1,K2
   39 DATA(K)=0
      GO TO 42
   40 CONTINUE
      BUFFER IN (1,1) (TABLE(1),TABLE(I1))
      ISTAT=UNIT(1)
      IF(ISTAT) 10,5,6
    6 PRINT 60
   60 FORMAT (* ERROR ON TERRAIN INPUT TAPE IGNORED*)
C         SEARCH (IN SEQUENCE) FOR PROPER TERRAIN REGION
   10 IF(TABLE(1).NE.ITREG) GO TO 11
      IPNT=IK*(IYCOR-1)+IXCOR
      ISHIFT=MOV.(MOD(IPNT,4)+1)
      IPNT=(IPNT-1)/4+2
      ITALT=AND(SHIFT(TABLE(IPNT),-ISHIFT),77778)
      TARGALT=FLOAT(ITALT)
   11 IF(TABLE(1).NE.REG) GO TO 60
C         INSERT ELEVATION DATA DIRECTLY IN REGION RECORD
      KT=2
      DO 15 K=K1,K2
      DATA(K)=TABLE(KT)
      KT=KT+1
   15 CONTINUE
   42 CONTINUE
C         WRITE BACK OUT AND DO NEXT REGION
      CALL WRITMS(12,DATA,K2,IREG)
    2 CONTINUE
    1 CONTINUE
      IDEXM1=INDEX-1
      PRINT 500, (LDATA(K),K=1,IDEXM1)
  500 FORMAT(*OCULTURE DATA FOR THE FOLLOWING REGIONS WAS OUTPUT: */
     1       * (IN FORM -REGION  NNOS-)* / (1X 10(I6,I5)) )
   55 CALL CLOSMS(12)
C           PASS PARAMETERS TO NEXT PROGRAM
   17 DLINK(1)=RANGE
      DLINK(2)=ALTITUDE
      DLINK(3)=X
      DLINK(4)=Y
      DLINK(5)=TARGALT
      LDATA(1)=NX
      LDATA(2)=NRSI
      LDATA(3)=ITRED
      LDATA(4) = IK
      LDATA(5)=ITERR
      BUFFER OUT(5,1) (DLINK(1),DLINK(5))
      IF(UNIT(6)) 18,20,20
   18 BUFFER OUT(6,1) (LDATA(1),LDATA(5))
      IF(UNIT(6)) 19,20,20
   19 REWIND 6
      PRINT 300,ITALT
  300 FORMAT(* ELEVATION AT LOCATION OF TARGET IS *,I4/
```

```
     +' RLAST SUCESSFUL END')
C
      STOP
  700 PRINT 701
  701 FORMAT(' ERROR-CULTURE RESOLUTION NOT EQUAL TO TERRAIN RESOL.')
      STOP
 20   PRINT 800
 800  FORMAT(' HIT EOF OR PARITY ON I/P TAPE')
      STOP
    5 PRINT 50, IEG
   50 FORMAT( ' END OF TERRAIN INPUT FILE REACHED ON REGION', I6)
      GO TO 25
      END
```

```
      PROGRAM  RSS8(INPUT,OUTPUT,TAPE4,TAPE12,TAPE6=0,TAPE20=0)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C          PROGRAM PLMS8: APPLY FLIGHT CONSTANTS AND GENERATE SWEEPS
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C          THE FOLLOWING IS AN INTERPRETATION OF VARIABLE USAGE.....
C
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C*****ARRAYS****
      DIMENSION ASIY(23467), DLINK(5)
      DIMENSION IT(60),IARRAY(50,50)
      INTEGER WORD(10)
      INTEGER SCNLNE(4000)
      INTEGER DATA(1000),NLINK(5)
      INTEGER ARRAY(50,50)
C DATA    - FEATURE CODE INTENSITY LEVEL (CONVERSION TABLE)
C SCNLNE  - SCAN LINE STORAGE FOR ONE ROW OF CULTURE INTENSITIES AND
C           CORRESPONDING HEIGHTS.
C II      - INTENSITY CODES FOR EACH FEATURE (ASSIGNMENT TABLE)
C IARRAY  - INTENSITY VALUES FOR EACH ELEMENT IN CULTURE REGION
C ARRAY   - HEIGHT VALUES AT EACH POINT IN THE TERRAIN REGION
C DLINK   - SINGLE RECORD AREA FOR PARAMETER PASSAGE, FL. PT.
C NLINK   - SINGLE RECORD AREA FOR PARAMETER PASSAGE, FX. PT.
C
C****SCALARS, FIXED POINT****
      INTEGER RET
      INTEGER XI,YI
C   THE CONTROLLING FACTOR IN THE DATA SCAN IS THE NUMBER OF ACTUAL
C    DATA POINTS AVAILABLE, IN THIS CASE FROM THE CULTURE FILE AT 156.25
C    FEET PER UNIT.  A GRID OF THIS RESOLUTION IS REFERRED TO AS THE
C    GROSS GRID TO DISTINGUISH IT FROM THE ARRAY LAYOUT WHICH DEFINES
C    THE REGION NUMBERS.
C NRX,NRY- REGION GRID LAYOUT NUMBERING LIMITS
C IBACK   - INTENSITY CODE FOR BACKGROUND OF PICTURE
C XI,YI   - RADAR LOCATION IN TERMS OF (GROSS) DATA GRID
C ID      - NUMBER OF ELEMENTS IN A LINE SWEEP. MAX IS 770 BECAUSE OVER
C           A RANGE OF 20NM, THE MAXIMUM NUMBER OF ARRAY ELEMENTS (ALONG
C           THE DIAGONAL) TIMES THE NUMBER OF REGIONS INVOLVED IS ABOUT
C           770 (24 REGIONS * 32 ELEMENTS EACH).
C IMIN    - STARTING SCAN ANGLE (+1)
C IFREQ   - NO. OF SWEEPS PER DEGREE
C IMAX    - ENDING SCAN ANGLE (+1)
C NREG    - REGION CODE FOR WHICH DATA IS CURRENTLY IN CORE
C REG     - REGION WHICH SWEEP LINE ELEMENT IS IN
C IANGLE  - SCAN ROTATION ANGLE COUNTER
C IA,IB   - ROW/COL OF CURRENT (GROSS) ELEMENT NUMBER UNDER BEAM SCAN
C KX,KY   - ROW/COL OF CURRENT ELEMENT UNDER BEAM SCAN WITHIN
C           A GIVEN CULTURE REGION.
C KKX,KKY- SAME AS KX,KY (TRANSFORMED TO TERRAIN REGION AT 3 TO 4 RATIO).
```

```
C
C****SCALARS, FLOATING POINT****
C X,Y     - RADAR LOCATION
C RANGE   - RADAR RANGE IN N. MILES
C CONV    - RADIANS PER DEGREE
C DCONV   - DISTANCE CONVERSION (PTS PER N.M.) FOR RANGE OVER GROSS GRID
C ALPHA   - STARTING ANGLE OF RADAR SWEEP-SCAN
C P       - CURRENT ANGLE OF SWEEP LINE
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      EQUIVALENCE (RANGE,DLINK(1)),(ALTIDE,DLINK(2))
      EQUIVALENCE (X,DLINK(3)), (Y,DLINK(4))
      EQUIVALENCE (NRX,NLINK(1)), (IFREQ,NLINK(3))
      EQUIVALENCE (ITEPR,NLINK(5)),(IK,NLINK(4))
C     NAME LIST /PARAMS/ IBACK,IFREQ,ALPH
C
C     ASSIGN INTENSITY DISPLAY CODES BY FEATURE
C
C
      DATA (IT(J),J=1,60)/1,13,3,2,3,3,1,4,12,14,0,10,15,5,3,15,0,15,
     +11,11,15,15,15,2,11,11,6,4,8,8,12,11,2,/*0,1,2,3,4,5,6,7,8,9,10,
     +11,12,13,1,,6*15/
      DATA IN2W/20/
C
      BUFFER IN (IN2W,1) (WORD(1),WORD(2))
      IF(UNIT(IN2W)) 33,111,111
   33 RESK=WORD(1)/304.8
      IKC=WORD(2)
C
      CALL OPENMS(12,MSTX,28467,0)
      IBACK=61
      CONV=ATAN2(0.,-1.)/180.
      DCONV=6076.11549/RESK
      ECONV=DCONV
C
C     FETCH RADAR PARAMETERS FROM PREVIOUS PROGRAM
C
      BUFFER IN(5,1) (DLINK(1),DLINK(5))
      IF(UNIT(6)) 32,115,115
   32 BUFFER IN(6,1) (NLINK(1),NLINK(5))
      IF(UNIT(6)) 34,115,115
   34 INTN=1
      ALPHA=0.
      IMAX=360*IFREQ
      XI=(X*ECONV+.5)+1 $YI=(Y*ECONV+.5)+1
C     ID=RANGE*DCONV
      ID=RANGE*DCONV+.5 $ ID=ID+1
      ID4=RANGE*DCONV/4.+.5 $ ID4=ID4+1
      PRINT 65, KZ,ID,LZ,ID4
      IL = IK/2
      IN=IK/4
      IF(ID.GT.2000) ID=2000
      NREG=0
      TARX = X*ECONV $ TARY = Y*ECONV
```

COPY AVAILABLE TO DDC DOES NOT PERMIT FULLY LEGIBLE PRODUCTION

```
      PRINT 600,IARX,IARY
  600 FORMAT(1X,'TARGET LOCATION X =',F8.2,'   Y =',F8.2,'  GRID INTERVA
     1LS')
      ERRX = IARX-(XI-1)
      ERRY = IARY-(YI-1)
      PRINT 601,ERRX,ERRY
  601 FORMAT('0TARGET ROUNDOFF ERROR.  ERRX=',F8.2,'  ERRY=',F8.2)
      PRINT 300, XI,YI,ID,ALPHA
  300 FORMAT('0NEAREST GRID INTERSECTION. RADAR AT COL=',I5, ' ROW=',I5/
 3001      '  BEAM DISTANCE (RANGE)= ', I7, ' GRID ELEMENTS'/
 3002      '  INITIAL SWEEP ANGLE OFFSET=', F6.1, ' DEGREES')
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C      GENERATE EACH SWEEP LINE IN TURN
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      DO 50 IANGLE=IMIN,IMAX
      P=(IANGLE-1)/FLOAT(IFREQ)
      THETA=(ALPHA+P)*CONV
      STH=SIN(THETA)$CTH=COS(THETA)
C + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
C        FILL EACH ELEMENT OF SWEEP LINE
C + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
C     DO 51 K=1,ID,2
      LZ=KZ=0 $ K=1 $ INCR=1
      IF(IANGLE.EQ.IMIN) PRINT 66
   66 FORMAT(//30X,'STARTED LOOP 51 FOR THE FIRST TIME'/)
   6F HALFX=.5
      HALFY=.5
      IF(CTH.LT.-.01) HALFX=-.5
      IA=XI+INT((K-1)*CTH+HALFX)
      IF(STH.LT.-.01) HALFY=-.5
      IB=YI+INT((K-1)*STH+HALFY)
      IANY=MOD(IA,IKC)
      IF(IANY.EQ.0) GO TO 800
      KX=IA-IA/IKC*IKC
  801 IANY=MOD(IB,IKC)
      IF(IANY.EQ.0) GO TO 803
      KY=IB-IB/IKC*IKC
  802 IF(IANX.EQ.0) IA=IA-1
      IF(IANY.EQ.0) IB=IB-1
      REG=1+IA/IKC+IB/IKC*NRX
      GO TO 504
  800 IAA=IA-1
      KX=IA-IAA/IKC*IKC
      GO TO 801
  803 IBB=IB-1
      KY=IB-IBB/IKC*IKC
      GO TO 802
C
C        SEE IF DATA IN CORE IS FOR THIS REGION
C
  504 IF(NREG.EQ.REG) GO TO 50
C
C        NO, SO GET READY FOR NEW REGION
```

COPY AVAILABLE TO DDC DOES NOT PERMIT FULLY LEGIBLE PRODUCTION

```
      C
      C     FILL ARRAY WITH BACKGROUND INTENSITY VALUES
      C
            DO 1 I=1,IKC
            DO 1 J=1,IKC
          1 IARRAY(I,J)=IBACK
      C     READ DATA FOR THIS REGION
            INR=REG
            CALL READMS(12    ,DATA,800,INR)
            NREG=REG
            ILIM=DATA(1)+1
      C        CHECK FOR VALIDITY
      C        CULTURE MAY BE MISSING WHERE TERRAIN EXISTS
            IF(DATA(1).EQ.0)GO TO 10
            IF(DATA(1).GE.500) GO TO 100
      C        UNPACK CULTURE DATA FROM REGION RECORD AND
      C     ENTER INTENSITY CODES INTO ARRAY BASED ON CULTURE STRIPS IN REGION
            DO 9 I=2,ILIM
            IYL=AND(SHIFT(DATA(I),-30),77B)
            IF(IYL.GT.IKC) GO TO 105
            IXL=AND(SHIFT(DATA(I),-24),77B)
            IFL=DATA(I).AND.77B
            IDXL=JLIM=AND(SHIFT(DATA(I),-18),77B)
            DO 8 J=1,JLIM
            JM1=J-1
            IF(IXL+JM1.GT.IKC) GO TO 115
      C
      C        KEEP FEATURE CODES, WILL TRANSLATE TO GRAY SHADES IN PROG 9
      C     IARRAY(IYL,IXL+JM1)=II(IFL)
            IARRAY(IYL,IXL+JM1)=IFL
      C
          8 CONTINUE
          9 CONTINUE
      C + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
      C        EXTRACT ELEVATION DATA FROM REGION (STARTS AFTER CULTURE)
      C + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
         10 CONTINUE
            IJ=ILIM+1
            IXL=0
            DO 20 IIX=1,IK
            IXL=IXL+1
            JJ=IJ*(IIX-1)
            DO 20 IIY=1,IK,4
            IIP=IJ+IIY/4+JJ
            ARRAY(IXL,IIY)=AND(SHIFT(DATA(IIP),-45),77777B)
            ARRAY(IXL,IIY+1)=AND(SHIFT(DATA(IIP),-30),77777B)
            ARRAY(IXL,IIY+2)=AND(SHIFT(DATA(IIP),-15),77777B)
            ARRAY(IXL,IIY+3)=DATA(IIP).AND.77777B
         20 CONTINUE
      C + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
      C        SELECT INTENSITIES AND HTS FROM LOADED ARRAYS THAT
      C        CORRESPOND TO THIS SCAN LINE ELEMENT.
      C + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
```

```
      50 CONTINUE
C     50 PNAP=WORD(1)/NLINK(2)
C        IF(PMAP.LE.1.) GO TO 53
C        KX=IA-(IK*((IA-1)/IK))
C        KY=IB-(IK*((IB-1)/IK))
C        GO TO 56
      53 KX=IA-(IKC*((IA-1)/IKC))
         KY=IB-(IKC*((IB-1)/IKC))
C        IF(PMAP.EQ.1.) GO TO 54
C     56 KKX=KX*PMAP+1
C        KKY=KY*PMAP+1
C        GO TO 58
C     54 KKX=KX
C        KKY=KY
C
      58 INZ=IARRAY(KY,KX)
C
C        IHT=ARRAY(KKY,KKX)
         IHT=ARRAY(KY,KX)
C        SCNLNE(K)=INZ$SCNLNE(K+2000)=IHT
C
C           1/4 DEGREE SCAN LINES WITH 25 METER SPACING IS TOO MUCH DATA.
C           SKIP EVERY OTHER POINT (INCR=2) TO GET 50 METER DATA. AT
C           BEGINNING OF LINE THOUGH (UP TO 4K RADIUS), DO NOT SKIP
C           POINTS (INCR=1) TO GET 25 METER DATA FOR LOWEST ALTITUDE REFERENCE
C           SCENE. WHEN DOING THIS, WRITE EVEN NUMBERED (FILL) POINTS AT
C           END OF SCANLINE ARRAY.
C
         IF(INCR.GT.1) GO TO 69
         IF((K-(K/2)*2).NE.0) GO TO 69
         LZ=LZ+1 $ SCNLNE(ID+LZ)=INZ $ SCNLNE(ID+LZ+1000)=IHT $ GO TO 62
      69 KZ=KZ+1 $ SCNLNE(KZ)=INZ $ SCNLNE(KZ+1000)=IHT
      62 IF(INCR.GT.1) GO TO 51
         IF((K+INCR).LT.ID4) GO TO 51
         IF((K+INCR).GT.ID4) GO TO 63
         IF(IABS(ID4-(ID4/2)*2).EQ.0) GO TO 51
      63 K=K-1 $ INCR=2
         IF(IANGLE.EQ.IMIN) PRINT 61, INCR,K
      61 FORMAT(//10X,'INCR CHANGED TO',I7,7X,'AT K =',I7)
      51 K=K+INCR
         IF(IANGLE.GT.IMIN) GO TO 64
         IF(K.LE.15) PRINT 67, K,INCR
         IF(IABS(K-(K/11)*11).EQ.0) PRINT 67, K,INCR
         IF(K.GE.1415) PRINT 67, K,INCR
      67 FORMAT(5X,'IN LOOP 51 K =',I7,7X,'INCR =',I7)
      64 IF(K.LE.ID) GO TO 60
C     51 CONTINUE
C                             FINISHED ONE SCAN LINE
         IF(IANGLE.EQ.IMIN) PRINT 65, KZ,ID,LZ,ID4
      65 FORMAT(////10X,'KZ =',I7,7X,'ID =',I7,7X,'LZ =',I7,7X,'ID4 =',I7
        .  ///)
C
C + + + + + + + + + + + + + + + + + + + + + + + + +
```

```
C        OUTPUT THE COMPLETED SCAN LINE
C * * * * * * * * * * * * * * * * * * * * * * * * * * *
      BUFFER OUT(4,1) (SCNLNE(1),SCNLNE(2000))
      IF(UNIT(4))752,119,119
  752 IZTEST=IABS(IANGLE-(IANGLE/50)*50)
      IF(IANGLE.LE.10) PRINT 753, IANGLE
      IF(IANGLE.GE.1455) PRINT 753, IANGLE
      IF(IZTEST.EQ.0) PRINT 753, IANGLE
  753 FORMAT(20X,'...  FINISHED SCANLINE',I7,'    ...')
   52 CONTINUE
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C        CLOSE OUT
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      REWIND 4
   55 REWIND 6
      PRINT 255
  255 FORMAT(//////30X,'*****  SUCCESSFULL END OF RSS 8  *****'//////)
      STOP
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C        DATA STORAGE OR I/O ERRORS
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C        TOO MANY CULTURE STRIPS FOR A REGION
  100 PRINT 200, DATA(1), REG
  200 FORMAT(' VALUE COUNT=',I6, ', IN ERROR FOR REGION',I6)
  101 PRINT 220,P
  220 FORMAT(' ....OCCURRED DURING SWEEP ANGLE OF', F6.1, ' DEGREES')
      STOP
C     ROW OR COLUMN INDEX OUT OF BOUNDS FOR REGION
  105 PRINT 205, IYL,    REG
  205 FORMAT( ' COL ELEMENT VALUE=',I6, ', IN ERROR FOR REGION', I6)
      GO TO 101
  110 PRINT 210, IXL  +JM1,REG
  210 FORMAT( ' ROW ELEMENT VALUE=',I6, ', IN ERROR FOR REGION', I6)
      GO TO 101
  115 PRINT 215
  215 FORMAT( ' EOF OR PARITY ERROR ON UNIT 6, FLMS8')
      GO TO 101
  119 PRINT 219
  219 FORMAT('EOF OR PARITY ERROR ON O/P FILE 4')
      GO TO 101
  111 PRINT 211, IN2W
  211 FORMAT(' INPUT ERROR ON UNIT ',I2)
      END
```

```
PROGRAM RADAR9      73/74   OPT=1                           FTN 4.6+42:          10/20/76  09.25.21        PAGE    1

      PROGRAM RADAR9(INPUT,OUTPUT,TAPE4,TAPE8,TAPE10,TAPE12,TAPE14,
     .         TAPE6=0,TAPE20=0,TAPE5=INPUT)
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C         PROGRAM   RSS9N   APPLY RADAR EFFECTS
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      COMMON/SWITCH/KOUNT,IDIST,ISHADE,SCALE,IRASTER
      COMMON /ARRAYS/ SR(1000),AVINS(1000),DATA(4000),ALT,REST,RESK
      COMMON /LARRAY/ PINS(1000) , EMTS(1000) , RET(64) , ATARG , ZBIAS
      COMMON / LIMITS / LIM10,LIM12,LIM14
      COMMON / GRAYLVL / INTNSTY(64) , I3
      DIMENSION ICOLOR(64),ICODE(64),ICOLOR(64)
      INTEGER DATA,ELEV(2000),INS(2000),ALT,R,WORD(10)
      EQUIVALENCE (INS,DATA(1)), (ELEV,DATA(1001))
      NAMELIST / PARAMS /  REFALT , LAYOVER , IRASTER , ZANGLE
      DATA IN2V/20/ , LAYOVER/0/ , IRASTER/301/ , ZANGLE/0.0/
      DATA ICODE/64*0/,ICOLOR/64*0/,I3/61/
C
C        CONVERT FEATURE CODES TO 6 BIT INTENSITIES
C     DATA INTNSTY / 04,52,12,38,12,12,14,16,48,56,
      DATA INTNSTY / 04,52,12,00,12,00,00,16,58,56,
C    1                00,40,62,20,12,62,00,52,44,40,
     1                00,60,62,20,04,62,00,52,44,40,
     2                62,62,62,38,44,44,24,16,32,32,
     3                48,44,38,62,62,62,62,52,62,62,
     4                00,06,09,12,15,18,21,24,27,30,
     5                33,36,39,42,45,48,51,54,57,62,
     6                20,52,62,62 /
C
C
C                  CALIBRATIONS FOR MICRO DEF
      DATA RET / 2.5,2.46,2.42,2.38,2.34,2.30,2.26,2.22,2.18,2.14,
     .   2.10,2.06,2.02,1.98,1.94,1.90,1.86,1.83,1.79,1.75,
     .   1.71,1.67,1.63,1.59,1.55,1.51,1.47,1.43,1.39,1.35,
     .   1.31,1.27,1.23,1.19,1.15,1.11,1.07,1.03,.99,.95,
     .   .91,.87,.83,.79,.75,.71,.67,.64,.60,.56,
     .   .52,.48,.44,.40,.36,.32,.28,.24,.20,.16,
     .   .12,.08,.04,.00 /
C
C ***** ARRAY AND VARIABLE USAGE *****
C
C DATA     - INPUT DATA RECORD COMBINING CULTURE AND TERRAIN HTS BY SWEEP
C            LINE
C ELEV     - THAT PART OF INPUT DATA FOR TERRAIN VALUES
C INS      - THE CULTURE INTENSITY PORTION OF THE INPUT DATA RECORD
C SR       - RADAR STRENGTH OF RETURN AT EACH PT, COMPUTED BY SUBPT
C            LAMBERT
C MM       - SWEEP ANGLE
C
      ZFACT = (63.0-00.0) / (RET(1)-RET(64))
      RET1=RET(1) $ ZBIAS=1000000.
      DO 52 I=1,64
```

```
   72 REF(I)=10**REF(I)
      BUFFER IN (IN2W,1) (WORD(1),WORD(2))
      IF(UNIT(IN2W)) 51,230,230
   51 RESK=WORD(1)/304.8
      KOUNT=0
      SCALE=1.0
      DO 8 I=1,1001
    8 AVINS(I) = 0
      CONV=ATAN2(0.,-1.)/180.
      BUFFER IN(5,1)(SR(1),SR(5))
      IF(UNIT(6)) 2,60,60
    2 CONTINUE
      READ  PARAMS
      IF((REFALT.LT.4000.).OR.(REFALT.GT.32000.)) GO TO 101
      SR(2)=REFALT
  101 SR(1)=3.64+5*SR(2)/6076.11549
      ALT=SR(2)+SR(5)
      ATARG = SR(2)
      PRINT 750, ATARG
  750 FORMAT(//////,20X,'*****   PROGRAM 9 TO MAKE',F9.0,'     FOOT REFERE
     .NCE SCENE   *****',//////)
      PRINT 50,SR(5)
   50 FORMAT('0ELEVATION OF TARGET ABOVE SEA LEVEL IS',F7.0,'  FEET')
      PRINT 7750, ALT
 7750 FORMAT('0  ELEVATION OF TARGET REFERENCE RADAR ABOVE SEA LEVEL IS'
     .  ,I7,'  FEET')
      IRASTER=(IRASTER+1)/2
      IF((IRASTER.LT.32).OR.(IRASTER.GT.1000)) IRASTER=151
      MM=(IRASTER*2)-1
      PRINT 4000, MM,MM
 4000 FORMAT('0    OUTPUT RASTER SIZE IS',I6,'    BY',I6)
      BUFFER IN(5,1) (DATA(1),DATA(5))
      IF(UNIT(6)) 3,60,60
    3 IFREQ=DATA(3)
      REST=DATA(2)/304.8
      IF(DATA(2).NE.WORD(1)) GO TO 240
C
C     IF(REFALT.GT.4000.) REST=REST*2
      REST=REST*2.
C
      RESK=REST $ PRINT 31, REST,RESK,IFREQ
   31 FORMAT('0  RESOLUTION (IN FEET) OF TERRAIN',F11.4,7X,'PLANIMETRY',
     .  F11.4,7X,'AND 1 OVER',I4,4X,'DEGREES DELTA THETA')
      LIM10=90*IFREQ          $ LIM12=LIM10*2 $ LIM14=LIM10*3
      IDIST=SR(1)*6076.11549/RESK+.5 $ IDIST=IDIST+1
      IF(IDIST.GT.1000) IDIST=1000
      PRINT 1,IDIST
    1 FORMAT('0RADAR RANGE IS ',I6,' INCREMENTS ')
C
      IMIN=1 $ IMAX=360*IFREQ  $ ITEMP=IDIST
      IF(LAYOVER.NE.0) GO TO 520
      PRINT 510 $ GO TO 7530
  510 FORMAT('0',30X,'LAYOVER USED IN THIS RUN')
```

```
  520 PRINT 525
  525 FORMAT('0',30X,'LAYOVER WAS *NOT* USED IN THIS RUN')
 7530 ZRES=3.6443 * ATARG / FLOAT(IFACTER-1) $ ZRESM=ZRES*.3048
      PRINT 485, ZRES, ZRESM
  485 FORMAT('0 RESOLUTION ALONG RADIAL LINES IS',F9.3,' FEET OR',
     .   F9.3,' METERS')
C        START RADAR BEAM SWEEP
      DO 100 MM=IMIN,IMAX
      IDIST=ITEMP
      THETA=(MM-1)/FLOAT(IFREQ)+ZANGLE
      THETA=THETA*CONV
C        INPUT SEQUENCE OF RECORDS MUST BE EXACTLY AS RLMS8 OUTPUT
      BUFFER IN(4,1) (DATA(1),DATA(4000))
      IF(UNIT(4)) 4,70,70
    4 IF(IDIST.EQ.IRASTER) GO TO 5
      CALL SIZE
    5 CONTINUE
      DO 7 I=1,IDIST
    7 ICODE(INTNSTY(DATA(I))+1)=ICODE(INTNSTY(DATA(I))+1) + 1
      CALL LAMBERT
      DO 6 R=1,IDIST
C
      J=INS(R) $ J=INTNSTY(J) $ RINS(R)=RET(J+1)+ZBIAS
      IF(INS(R).EQ.13) RINS(R)=(RINS(R)-ZBIAS)*SR(R)
      IF(INS(R).EQ.14) RINS(R)=(RINS(R)-ZBIAS)*SP(R)
      RINSE=RINS(R)
      IF(RINSE.GE.ZBIAS) RINSE=RINSE-ZBIAS
      IF(RINSE.LT.1.) RINSE=1.
      J=(RET1-ALOG10(RINSE)) * ZFACT + .5
      IF(J.GT.63) J=63
      IF(J.LT.0) J=0
      ICOLOR(J+1)=ICOLOR(J+1)+1
    6 CONTINUE
C
C     COMPUTE ALTITUDE LAYOVER EFFECT
C
      IF(LAYOVER.NE.0) GO TO 530
      CALL ALTLAV
C
  530 DO 410 R=1,IDIST
      BIAS=0.0
      IF(RINS(R).GE.ZBIAS) BIAS=ZBIAS
      RINSE=RINS(R)-BIAS
      IF(RINSE.LT.1.0) RINSE=1.0
      INS(R)=(RET1-ALOG10(RINSE)) * ZFACT + .5
      IF(INS(R).GT.63) INS(R)=63
      IF(INS(R).LT.0) INS(R)=0
      AVINS(R)=AVINS(R)+INS(R)
      ICOLO2(INS(R)+1)=ICOLO2(INS(R)+1)+1
      IF(BIAS.GE.ZBIAS) INS(R)=OR(INS(R),100B)
  410 CONTINUE
C
C     REFORMAT DATA FOR RECTANGULAR COORDINATES
```

```
      C     SAVE PROCESSED SWEEPS FOR RASTER CONVERSION
            CALL REFRMT(THETA)
        100 CONTINUE
      C
      C     COMPUTE AVERAGE INTENSITIES
      C
            PRINT 210
        210 FORMAT('0RL459 SUCCESSFUL END, DISPLAY FILE COMPLETE')
        211 ENDFILE 3 $ ENDFILE 10 $ ENDFILE 12 $ ENDFILE 14
            DO 9 I=1,IDIST
          9 AVINS(I) = AVINS(I)/(360.*IFREQ)
            PRINT 160
        160 FORMAT('  INPUT STATISTICS')
            DO 150 I=1,64
            IF(ICODE(I).EQ.0) GO TO 150
            PRINT 151,ICODE(I),I-1
        151 FORMAT(I10,3X,'PIXELS WITH COLOR CODE',I3)
        150 CONTINUE
       7700 PRINT 161
        161 FORMAT(' OUTPUT STATISTICS')
            DO 215 I=1,64
            PRINT 216,ICOLOR(I),I-1,ICOLO2(I),I-1
        216 FORMAT(I9,'  LAMBERT PIXELS WITH COLOR CODE',I3,10X,I9,2X,'ALTLAV
           .PIXELS WITH COLOR CODE',I3)
        215 CONTINUE
            PRINT 217,ISHADE
        217 FORMAT(I9,'  PIXELS WERE ASSIGNED TO THE SHADOW')
            PRINT 218, (AVINS(I),I=1,1000)
        218 FORMAT(' AVERAGE OF INTENSITIES'/50(20F5.1/))
            STOP
         60 PRINT 305 $ GO TO 211
        305 FORMAT('0RL459 - ERROR OR EOF ON UNIT6')
         70 PRINT 310 $ GO TO 211
        310 FORMAT('0RL459 - EOF OR PARITY ERROR IN I/P FILE 4')
        230 PRINT 231, INOW $ STOP
        231 FORMAT(' INPUT ERROR ON UNIT ',I2)
        240 PRINT 241, RESK , REST
        241 FORMAT('0 ERROR, CULTURE RESOLUTION =',F8.3,'  NOT EQUAL TO TERRAI
           .N RESOLUTION =',F8.3)
            STOP $ END
```

```
      SUBROUTINE SIZE
      COMMON /LAYARY/ PINS(1000) , ENTS(1000) , PET(64) , ATARG , ZBIAS
      COMMON/SWITCH/KOUNT,IDIST,ISHADE,SCALE,IRASTER
      COMMON /ARRAYS/ SR(1000),AVINS(1000),DATA(4000),ALT,REST,PESK
      COMMON / GRAYLVL / INTNSTY(64) , IB
      INTEGER DATA,ELEV(2000),INS(2000),ALT,R
      INTEGER INSV(1000) , ELESV(1000)
      EQUIVALENCE (INS,DATA(1)), (ELEV,DATA(1001))
C
C        IF THIS IS THE 4000 FOOT LOW ALTITUDE REFERENCE SCENE, THEN GET THE
C        EXTRA FILL POINTS FROM END OF INS AND ELEV ARRAYS TO DOUBLE
C        THE INPUT RESOLUTION (SEE PROGRAM 8)
C
C     IF(ATARG.GT.4000.) GO TO 290
      IF(ATARG.GT.1000.) GO TO 290
      J=3.6445*4000./(REST*2.)+1.5 $ I=(J-1)*2+1 $ ISTOP=I-1
  200 INS(I)=INS(J) $ ELEV(I)=ELEV(J)
      IF(J.LE.2) GO TO 210
      J=J-1 $ I=I-2 $ GO TO 200
  210 J=3.6445*32000./(REST*2.)+2.5 $ I=2
  220 INS(I)=INS(J) $ ELEV(I)=ELEV(J)
      IF(I.GE.ISTOP) GO TO 290
      I=I+2 $ J=J+1 $ GO TO 220
C
  290 SCALE=FLOAT(IDIST-1)/FLOAT(IRASTER-1)
      ELESV(IRASTER)=ELEV(IDIST) $ ELESV(1)=ELEV(1)
      INSV(IRASTER)=INS(IDIST) $ INSV(1)=INS(1)
      ISTOP=IRASTER-1 $ IE=ISTOP
      DO 100 I=2,ISTOP
      RSCALE=SCALE*(IE-1)+1 $ J=RSCALE $ RSCALE=RSCALE-J
      ELESV(IE)=ELEV(J)+RSCALE*(ELEV(J+1)-ELEV(J))
      IF(SCALE-1.0) 10,70,60
   10 IF(RSCALE-0.5) 20,30,40
   20 IFC=INS(J) $ GO TO 50
   30 IFC=INS(J) $ IFCPLS=INS(J+1)
      IF(IFCPLS.EQ.IB) IFCPLS=IFC
      IF(IFC.EQ.IB) IFC=IFCPLS
      IF(IFC.EQ.IFCPLS) GO TO 50
      IF(INTNSTY(IFC)-INTNSTY(IFCPLS)) 50,50,31
   31 IFC=IFCPLS $ GO TO 50
   40 IFC=INS(J+1)
   50 INSV(IE)=IFC $ GO TO 100
   60 RSCALE=SCALE*(IE+.5-1.)+1. $ JPLS=RSCALE+.5
      RSCALE=SCALE*(IE-.5-1.)+1. $ J=RSCALE+.5 $ IFC=IB
   61 IFCPLS=INS(J)
      IF(IFCPLS.EQ.IB) GO TO 63
      IF(IFC.EQ.IB) IFC=IFCPLS
      IF(INTNSTY(IFC)-INTNSTY(IFCPLS)) 63,63,62
   62 IFC=IFCPLS
   63 IF(J.GE.JPLS) GO TO 64
      J=J+1 $ GO TO 61
   64 INSV(IE)=IFC
  100 IE=IE-1
```

```
      DO 110 I=1,IRASTER
      INS(I)=INSV(I)
  110 ELEV(I)=ELESV(I)
   70 IDIST=IRASTER $ RETURN $ END
```

```
      SUBROUTINE LAMBERT
C
C     INPUT
C       ELEV     ARRAY CONTAINING 770 ELEVATIONS ALONG SWEEP LINE
C                IN FEET ABOVE REFERENCE PLANE
C       ALT      AIRCRAFT ALTITUDE IN FEET ABOVE REFERENCE PLANE
C     OUTPUT
C       SRT      STRENGTH OF RETURN,0.0 TO 1.0
C     INTERNAL VARIABLES
C       MT       SLOPE OF TERRAIN AT THE POINT OF INTEREST
C       MV       SLOPE OF VECTOR FROM AIRCRAFT TO TERRAIN POINT
C
      COMMON/SWITCH/KOUNT,IDIST,ISHADE,SCALE,IRASTER
      COMMON /ARRAYS/ SR(1000),AVINS(1000),DATA(4000),ALT,REST,RESK
      INTEGER DATA,ELEV(2000),INS(2000),ALT,R
      REAL MV , MT
      EQUIVALENCE (INS,DATA(1)), (ELEV,DATA(1001))
      DATA ZEXAG / 1.0 /
C
      SHADOW=.0
C     ADJUST FOR 10 NAUTICAL MILE RADIUS
      RES=SCALE*REST $ TOP=ALT-ELEV(1)
      ZEXAG=2.0
      R=2
    2 MV=(ELEV(R)-ALT)/((R-1)*RES)
      IF(MV.LT.-1.428) MV=-1.428
      IF(MV.GT.-.46631) MV=-.46631
      MT=(ELEV(R+1)-ELEV(R-1))/(2.*RES)
      IF(MV.GE.MT) GO TO 3
C
C     LAMBERTS LAW
C     A=(MV*MV)+1
C     B=(MT*MT)+1
C     SRT=(MT-MV)/SQRT(A*B)
C     IR = R
C     RD=140
C     IF(R.LE.RD) IR=RD
C     X = IR/500.
C     X=1.
C     SR(R)=X*SRT
C
C
C  STRENGTH OF RETURN IS SIN(ALPHA+BETA)/(COS(ALPHA)*SIN(BETA))
C  THEN ADD CUBIC POWER TERM FOR ANTENNA PATTERN
C
      MT=MT*ZEXAG
      SR(R)=-MT/MV+1.0 $ SR(R)=SR(R)*((TOP/(ALT-ELEV(R)))**3)
C
C
C     SHADOW APPROPRIATE AREAS
      IF(R.GE.IDIST-1) GO TO 5
      R=R+1
      GO TO 2
```

```
C       ENTER SHADOW
      3 SR(R)=SHADOW
        ISHADE=ISHADE+1
        IF(R.GE.IDIST-1) GO TO 5
        SLOPE = MV
        RP=R
        IF(MV.LE.-1.428) RP=(ALT-ELEV(R))/(RES*1.428)
        IF(MV.GE.-.4663) RP=(ALT-ELEV(R))/(RES*.4663)
      6 R=R+1
        RP=RP+1
C       TEST FOR END OF SHADOW
        AR=(SLOPE*RP*RES)+ALT
        IF(AR.GT.ELEV(R)) GO TO 7
        GO TO 2
C       REMAIN IN SHADOW
      7 SR(R)=SHADOW
        ISHADE=ISHADE+1
        IF(R.GE.IDIST-1) GO TO 5
        GO TO 6
C       FILL IN END POINTS OF SWEEP
      5 SR(1)=SR(2)
        SR(IDIST)=SR(IDIST-1)
        RETURN
        END
```

```
      SUBROUTINE ALTLAV
      COMMON/SWITCH/KOUNT,IDIST,ISHADE,SCALE,IRASTER
      COMMON /ARRAYS/ SR(1000),AVINS(1000),DATA(4000),ALT,REST,RESK
      COMMON /LAVARY/ RINS(1000) , ENTS(1000) , RET(64) , ATARG , ZBIAS
      INTEGER DATA,ELEV(2000),INS(2000),ALT,R
      INTEGER R1, R2, R3, R11
      INTEGER RBZ,RZ,RZZ
      EQUIVALENCE (INS,DATA(1)), (ELEV,DATA(1001))
      DO 2 R=1,1000
    2 ENTS(R) = 0
      ENTS1=RET(1) $ RES=SCALE*REST $ RFALSQ=ATARG*ATARG
      R2=(ATARG*2.144507/RES) + .5 $ R3=(ATARG*0.70020754/RES) + .5
      SHOWIN=RET(64)
      RZ=0 $ ISHFLG=0
      DO 3 R=1,IDIST
C     DO NOT SHIFT SHADOW REGION
      IF(SR(R).GT.0.0) GO TO 13
      RBZ=RZ+1 $ ISHFLG=13
      IF(RBZ.GT.IDIST) RBZ=IDIST
      IF(RBZ.EQ.1) ENTS(1)=SHOWIN
      IF(R.LT.IDIST) GO TO 3
      R1=R11=IDIST+1 $ GO TO 30
   13 NN=R-1
      IF(NN.GT.R2) NN=R2
      IF(NN.LT.R3) NN=R3
      RPM=NN*RES
      RP=RPM*RPM + (ALT-ELEV(R))**2 - RFALSQ
      IF(RP.GE.0.0) GO TO 10
      R1=R11=1 $ PR1=1.0 $ PR11=0.0 $ GO TO 20
   10 X=SQRT(RP)
      DX = X - RPM
      IXE=-1 $ DX=DX/RES $ IDX=DX
      IF(DX.GE.0.0) IXE=+1
      R1=IDX+R $ R11=R1+IXE $ PR11=(DX-IDX)*IXE $ PR1=1.0-PR11
      IF (R1.GT.IRASTER) GO TO 30
      IF (R1 .LE. 0) R1=1
      IF(R11.LE.0) R11=1
C     ADD THE INTENSITIES
   20 BIAS=0.0
      IF(RINS(R).GE.ZBIAS) BIAS=ZBIAS
      ENTS(R1)=(RINS(R)-BIAS)*PR1+ENTS(R1)
      IF(ENTS(R1).LT.ZBIAS) ENTS(R1)=ENTS(R1)+BIAS
      ENTS(R11)=(RINS(R)-BIAS)*PR11+ENTS(R11)
      IF(ENTS(R11).LT.ZBIAS) ENTS(R11)=ENTS(R11)+BIAS
   30 RZ=R1
      IF(R11.GT.R1) RZ=R11
      IF(ISHFLG.EQ.0) GO TO 3
      ISHFLG=0 $ RZZ=R11-1
      IF(R1.LT.R11) RZZ=R1-1
      IF(RBZ.LT.1) RBZ=1
      IF(RZZ.GT.IDIST) RZZ=IDIST
      IF(RBZ.GT.RZZ) GO TO 3
      DO 35 I=RBZ,RZZ
```

```
   35 ENTS(I)=ENTS(I)+SHOWIN
    3 CONTINUE
      ISHFLG=0 $ I=0 $ J=1 $ R=1
      IF(ENTS(1).GE.SHOWIN) GO TO 45
   42 R=R+1
      IF(ENTS(R).LT.SHOWIN) GO TO 42
      ENTS(1)=ENTS(R)
   45 DO 4 R=1,IDIST
      IF(ENTS(R).GE.SHOWIN) GO TO 50
      I=J $ ISHFLG=13
      IF(R.LT.IDIST) GO TO 4
      AVGIN=ENTS(I) $ RBZ=I+1 $ RZZ=IDIST $ GO TO 54
   50 RINS(R)=ENTS(R) $ J=R
      IF(ISHFLG.EQ.0) GO TO 4
   53 ISHFLG=0 $ RBZ=I+1 $ RZZ=J-1 $ BIAS=0.0
      IF(ENTS(I).GE.ZBIAS) BIAS=ZBIAS
      AVGIN=ENTS(I)-BIAS
      IF(ENTS(J).LT.ZBIAS) GO TO 153
      AVGIN=(AVGIN+ENTS(J)-ZBIAS)/2. $ AVGIN=AVGIN+ZBIAS $ GO TO 54
  153 AVGIN=(AVGIN+ENTS(J))/2. + BIAS
   54 IF(RBZ.GT.RZZ) GO TO 4
      DO 55 I=RBZ,RZZ
C  55 RINS(I)=AVGIN
   55 RINS(I)=SHOWIN
    4 CONTINUE
      RETURN $ END
```

```
      SUBROUTINE REFRMT(THETA)
      COMMON/SWITCH/KOUNT,IDIST,ISHADE,SCALE,IRASTER
      COMMON /ARRAYS/ SR(1000),AVINS(1000),DATA(4000),ALT,REST,RESK
      COMMON / LIMITS / LIM10,LIM12,LIM14
      INTEGER DATA,ELEV(2000),INS(2000),ALT,R
      INTEGER OUT(3)
      EQUIVALENCE (INS,DATA(1)), (ELEV,DATA(1001))
C
C  THIS ROUTINE PERFORMS THE POLAR-TO-RECTANGULAR CONVERSION FOR THE
C   RADAR SWEEPS TO RASTER SCANS.
C
C    INPUT - IS A SWEEP VECTOR OF INTENSITIES AND THE ROTATION ANGLE IN
C            RADIANS.
C   OUTPUT - IS SIMPLY THE ROW/COL INDICES FOR EACH POINT INTENSITY IN
C            THE SWEEP VECTOR (PACKED BOTH IN THE SAME WORD).  THESE ARE
C            STORED IN THE SECOND 'HALF' OF THE INPUT VECTOR SPACE.
C   OUTPUT INDICES ARE BASED ON AN ARBITRARY 770 X 770 GRID WITH AN
C   ASSUMED RADAR LOCATION AT GRID CENTER=(770,771), AND ORIGIN AT NW
C    CORNER.
C
      STH=SIN(THETA) $ CTH=COS(THETA) $ RADIUS=IDIST-1
      T=ABS(STH)
      IF(ABS(CTH).GT.ABS(STH)) T=ABS(CTH)
      IDIS=IDIST/T+.5
C
C        COMPUTE EACH COORDINATE FOR EACH POINT INTENSITY ALONG THE
C         SWEEP
C
      NMAX=(IRASTER*2)-1 $ KOUNT=KOUNT+1         $ IUNIT=3
      IF(KOUNT.GT.LIM10)  IUNIT=10
      IF(KOUNT.GT.LIM12)  IUNIT=12
      IF(KOUNT.GT.LIM14)  IUNIT=14
      IXS=IYS=IRASTER $ IGS=62
C     DO 40 R=1,IDIST
C     D=R-1
      DO 40 I=1,IDIS
      D=(I-1)*T $ R=I*T+.5
      XC=D*CTH $ YC=D*STH $ XR=RADIUS+XC $ YR=RADIUS-YC
      IXR=XR+.5 $ IYR=YR+.5 $ IXR=IXR+1 $ IYR=IYR+1
      IF(IXR.EQ.IXS) GO TO 10
      IF(IYR.EQ.IYS) GO TO 20
      GO TO 50
   10 IZ=IYR-IYS
      IF(IABS(IZ).NE.2) GO TO 50
      IZ=IZ/2 $ IYS=IYS+IZ $ GO TO 25
   20 IZ=IXR-IXS
      IF(IABS(IZ).NE.2) GO TO 50
      IZ=IZ/2 $ IXS=IXS+IZ
   25 OUT(1)=NMAX*(IYS-1)+IXS
      OUT(2)=(AND(DATA(R),77B)+AND(IGS,77B))/2
      IF((DATA(R).GE.100B).OR.(IGS.GE.100B)) OUT(2)=OR(OUT(2),100B)
      WRITE(IUNIT) (OUT(IZEE),IZEE=1,2)
   50 IXS=IXR $ IYS=IYR $ IGS=DATA(R)
```

```
      OUT(1)=NMAX*(IYR-1)+IXR $ OUT(2)=DATA(R)
      WRITE(IUNIT) (OJT(IZEE),IZEE=1,2)
   31 FORMAT('PARITY ERROR ON OUTPUT TAPE'/'THETA=',F5.1)
   40 CONTINUE
      RETURN
      END
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

MOUNT(VSN=PK0011,SN=ZRADAR2)
SETNAME(ZRADAR2)
ATTACH(TAPE3,SORT,ID=ET77878,CY=01)
ATTACH(TAPE10,SORT,ID=ET77878,CY=02)
ATTACH(TAPE12,SORT,ID=ET77878,CY=03)
ATTACH(TAPE14,SORT,ID=ET77878,CY=04)
REQUEST(ZZZZZ1A,SN)
REQUEST(ZZZZZ1B,SN)
REQUEST(ZZZZZ1C,SN)
REQUEST(ZZZZZ1D,SN)
REQUEST(ZZZZZ1E,SN)
REQUEST(ZZZZZ1F,SN)
REQUEST(TAPE2,SN)
REQUEST(TAPE11,SN)
REQUEST(TAPE13,SN)
REQUEST(TAPE15,SN)
REQUEST(TAPE27,*PF,SN)
FILE(TAPE3,BT=K,RB=31,RT=F,FL=20,MBL=620)
FILE(TAPE2,BT=K,RB=31,RT=F,FL=20,MBL=620)
SORTMRG.
FILE(TAPE10,BT=K,RB=31,RT=F,FL=20,MBL=620)
FILE(TAPE11,BT=K,RB=31,RT=F,FL=20,MBL=620)
SORTMRG.
FILE(TAPE12,BT=K,RB=31,RT=F,FL=20,MBL=620)
FILE(TAPE13,BT=K,RB=31,RT=F,FL=20,MBL=620)
SORTMRG.
FILE(TAPE14,BT=K,RB=31,RT=F,FL=20,MBL=620)
FILE(TAPE15,BT=K,RB=31,RT=F,FL=20,MBL=620)
SORTMRG.
FILE(TAPE27,BT=K,RB=31,RT=F,FL=20,MBL=620)
SORTMRG.
PURGE(RADAR,FINAL,ID=ET77878,CY=01)
CATALOG(TAPE27,FINAL,ID=ET77878,CY=01,AC=PLOTPE,RP=999)
TRANSF(ETRST)
EXIT.
AUDIT(SN=ZRADAR2)
SORT,VAR=DISK
BYTESIZE,60
FILE,SORT=TAPE3,OUTPUT=TAPE2
FIELD,ROW(1.1,1,INTEGER)
KEY,ROW(A)
END
SORT,VAR=DISK
BYTESIZE,60
FILE,SORT=TAPE10,OUTPUT=TAPE11
FIELD,ROW(1.1,1,INTEGER)
KEY,ROW(A)
END
SORT,VAR=DISK
BYTESIZE,60
FILE,SORT=TAPE12,OUTPUT=TAPE13
FIELD,ROW(1.1,1,INTEGER)
KEY,ROW(A)
END
```

```
SORT,VAR=DISK
BYTESIZE,60
FILE,SORT=TAPE14,OUTPUT=TAPE15
FIELD,ROW(1.1,1,INTEGER)
KEY,ROW(A)
END
MERGE
BYTESIZE,50
FILE,INPUT=TAPE2,TAPE11,TAPE13,TAPE15,OUTPUT=TAPE27
FIELD,ROW(1.1,1,INTEGER)
KEY,ROW(A)
END
```

```
      PROGRAM RSS10(INPUT,OUTPUT,TAPE1,TAPE2)
C
C     PROGRAM READS IN 2 WORD ARRAY IA, FIRST WORD IS TOTAL PIXEL ADDRESS
C     SECOND WORD IS GRAY SHADE TO PUT THERE. EACH LINE IS NPOINTS LONG
C     TOTAL OF NLINES IN PICTURE.
C
      COMMON / STUFF / BUF(2040),IRASTER,NRADIUS,ISKIP,IBACK6,NN(64)
      INTEGER BACK,IA(2),OUT,BUF,IFIL(10),ICOLOR(64)
      EQUIVALENCE (INWD1,IA(1)),(INWD2,IA(2))
      NAMELIST / PARAMS / IRASTER,IBLOWUP,IOFX,IOFY,ISKIP
      DATA IN,OUT,IRASTER,ISKIP / 1,2,301,1 /
      DATA IBLOWUP,IOFX,IOFY,NREC / 5,0,0,0 /
      DATA BACK/77777777777777777777B/
      DATA IFIL / 77B,7777B,777777B,77777777B,7777777777B,777777777777B,
     . 77777777777777B,7777777777777777B,777777777777777777B,003 /
      DATA ICOLOR,NN / 64*0,64*0 /
      CALL RANSET(53741)
      IBACK6=AND(BACK,77B)
C
C        READ IN DESIRED RASTER SIZE
      READ PARAMS
      IF((IRASTER.LT.63).OR.(IRASTER.GT.1999)) IRASTER=301
      NRADIUS=(IRASTER+1)/2 $ NLINES=(NRADIUS*2)-1 $ NPOINTS=NLINES
      PRINT 25, IRASTER,NRADIUS,NPOINTS,NLINES,ISKIP
   25 FORMAT('1  IRASTER',I7,5X,'NRADIUS',I7,5X,'NPOINTS',I7,5X,
     .  'NLINES',I7,5X,'ISKIP',I7,////)
C
C        TOTAL NUMBER OF CDC 60 BIT WORDS FOR ONE PRINT LINE
C        AND NUM BYTES TO FILL IN LAST BUFFER WORD
      IF(IBLOWUP.LT.1) IBLOWUP=1
  159 ILIMIT=IRASTER*IBLOWUP
      IF((ILIMIT.LE.2040).AND.(ILIMIT.GE.1)) GO TO 160
      IBLOWUP=IBLOWUP-1 $ GO TO 159
  160 NWORDS=(ILIMIT-1)/10+1 $ NFB=NWORDS*10-ILIMIT
      IF(NFB.LE.0) NFB=10
      DO 450 I=1,2040
  450 BUF(I)=BACK
      IF(IABS(IOFX).GT.IBLOWUP/2) IOFX=0
      IF(IABS(IOFY).GT.IBLOWUP/2) IOFY=0
C
C        IF Y OFFSET IS +, PUT IN FILL RECORDS AT TOP
      IOFYT=IOFY
      IF(IOFY.LE.0) GO TO 39
      IOFYT=0
   30 IF(NREC.GE.IOFY) GO TO 39
      BUFFER OUT(OUT,1) (BUF(1),BUF(NWORDS)) $ NREC=NREC+1
      IF(UNIT(OUT)) 30,31,31
   31 GO TO 30
   39 CONTINUE
C
C        START AT LINE 1. DEFAULT FOR UNFILLED PIXELS IS BACK
      LINE=1 $ INWD1=1 $ INWD2=IBACK6
C
```

```
      C       FILL ENTIRE OUTPUT ARRAY WITH BACKGROUND INTENSITY
         10 DO 11 I=1,IRASTER
         11 BUF(I)=IBACK6
      C
      C       KOUNT IS MAXIMUM TOTAL PIXEL ADDRESS (TPA) FOR THIS LINE
            KOUNT=LINE*NPOINTS$IEOF=0
      C
      C       SUBTRACTING IOFSET FROM TPA = LOCATION ALONG THIS LINE
            IOFSET=KOUNT-NPOINTS
      C
      C       FIRST ENTRIES FOR THIS LINE
            NUM=INWD1 $ IGS=INWD2
      C
      C       READ IN A TPA AND INTENSITY
         12 READ(IN) (IA(IZEE),IZEE=1,2)
            IF(EOF(IN)) 90,2015
       2015 IF(IOCHEC(IN)) 13,15
         13 PRINT 14 $ GO TO 15
         14 FORMAT(//,10X,*INPUT PARITY ERROR*)
      C
      C       IF AN END OF FILE, DUMP OUTPUT BUFFER AND STOP
         90 IF(INWD2.EQ.77B) INWD2=76B
            IF(INWD2.EQ.177B) INWD2=176B
            IEOF=13 $ GO TO 40
      C
      C       IS THIS TPA ON NEXT LINE. IF SO, WRITE PRESENT LINE
         15 IF(INWD2.EQ.77B) INWD2=76B
            IF(INWD2.EQ.177B) INWD2=176B
            IF(INWD1.GT.KOUNT) GO TO 40
      C
      C       NEW TPA OR ANOTHER INTENSITY OF CURRENT TPA. IF MORE THAN ONE
      C       INTENSUTY FOR A TPA, USE BRIGHEST (SMALLEST NUMERICALLY)
            IF(INWD1.NE.NUM) GO TO 16
      C
      C       IS NEW INTENSITY BRIGHTER
            IF((INWD2.LT.100B).AND.(IGS.GE.100B)) GO TO 12
            IF((INWD2.GE.100B).AND.(IGS.LT.100B)) GO TO 89
            IF(INWD2.GT.IGS) GO TO 12
      C
      C       STORE BRIGHTEST
         89 NUM=INWD1 $ IGS=INWD2 $ GO TO 12
      C
      C       GET ADDRESS ALONG THIS LINE AND PUT IN OUTPUT ARRAY
         16 NUMBER =NUM-IOFSET $ BUF(NUMBER)=AND(IGS,77B)
      C
      C       SAVE TPA AND INTENSITY JUST READ IN
            NUM=INWD1 $ IGS=INWD2 $ GO TO 12
      C
      C       PUT IN LAST PIXEL IN THIS LINE
         40 NUMBER=NUM-IOFSET $ BUF(NUMBER)=AND(IGS,77B)
      C
      C       SAVE JUST READ ONES AND SCAN FOR MISSING PIXELS
            NUM=INWD1 $ IGS=INWD2
```

```
      DO 500 I=1,NPOINTS
      J=BUF(I) + 1
  500 ICOLOR(J)=ICOLOR(J) + 1
      CALL SCAN
C
C
C        PACKS 1 PIXEL PER WORD ARRAY INTO 10 PIXELS PER WORD
C        PRIOR TO WRITING ON TAPE
C
C        ENLARGE THE LINE BY 'BLOWUP' NUMBER OF TIMES AND
C        PUT IN THE X OFFSET FOR FINE TUNING RASTER
      K=IRASTER $ J=ILIMIT+1
  100 DO 105 I=1,IBLOWUP
      J=J-1
  105 BUF(J)=BUF(K)
      IF(K.LE.1) GO TO 110
      K=K-1 $ GO TO 100
  110 IF(IOFX) 111,120,116
C
C        SHIFT RASTER TO RIGHT AND FILL ON THE LEFT
  111 J=ILIMIT+1
  112 J=J-1 $ BUF(J)=BUF(J+IOFX)
      IF((J+IOFX).GT.1) GO TO 112
      J=-IOFX
      DO 113 I=1,J
  113 BUF(I)=IBACKG
      GO TO 120
C
C        SHIFT RASTER TO THE LEFT AND FILL ON THE RIGHT
  116 J=ILIMIT-IOFX
      DO 117 I=1,J
  117 BUF(I)=BUF(I+IOFX)
      J=J+1
      DO 118 I=J,ILIMIT
  118 BUF(I)=IBACKG
  120 K=ILIMIT
      DO 700 J=1,K
      IWORD=(J-1)/10+1 $ IBYTE=J-(IWORD-1)*10 $ IBITPT=(IBYTE-1)*6+1
      IWRDPT=BUF(IWORD) $ LENGTH=6 $ IBITEX=55 $ IWRDEX=BUF(J)
C     SUBROUTINE BYTE(IBITEX,LENGTH,IWRDEX,IBITPT,IWRDPT)
C  A BYTE,LENGTH NUMBER OF BITS LONG (1 -60),IS EXTRACTED FROM
C  IWRDEX, STARTING AT BIT NUMBER IBITEX (BIT 1 MSB OF CDC WORD,
C  BIT 60 LSB OF CDC WORD). THIS BYTE IS PUT IN IWRDPT, STARTING AT
C  BIT NUMBER IBITPT. USES LEFT CIRCULAR WRAP AROUND.
C  CHECK FOR VALID VALUES
      IF((IBITEX.LT.1).OR.(IBITEX.GT.60)) GO TO 790
      IF((IBITPT.LT.1).OR.(IBITPT.GT.60)) GO TO 790
      IF((LENGTH.LT.1).OR.(LENGTH.GT.60)) GO TO 790
      ITEMP=1B$MASK=1B
      IDO=LENGTH-1
      IF(IDO.LE.0) GO TO 710
      DO 75 I=1,IDO
      MASK=SHIFT(MASK,1)
```

```
   75 MASK=OR(MASK,ITEMP)
  710 ITEMP=SHIFT(IWRDEX,(LENGTH+IBITEX-1))
      ITEMP=AND(ITEMP,MASK)
      IDO=SHIFT(IWRDPT,(LENGTH+IBITPT-1))
      IDO=AND(IDO,COMPL(MASK))
      IDO=OR(IDO,ITEMP)
      IWRDPT=SHIFT(IDO,(61-LENGTH-IBITPT))
  700 BUF(IWORD)=IWRDPT
C
C        FILL REMAINING BYTES WITH BACKGROUND INTENSITY
      BUF(NWORDS)=OR(BUF(NWORDS),IFIL(NFB))
C
C        WRITE OUT A LINE*IBLOWUP OF PICTURE
      ITEMP=IBLOWUP
C
C        IF IOFY IS -, WRITE FEWER TIMES AT FIRST AND FILL AT BOTTOM
      IF(LINE.EQ.1) ITEMP=IBLOWUP+IOFYT
C
      DO 145 I=1,ITEMP
      IF(NREC.GE.ILIMIT) GO TO 145
      BUFFER OUT(OUT,1)(BUF(1),BUF(NWORDS)) $ NREC=NREC+1
      IF(UNIT(OUT))145,45,41
   41 PRINT 42,NREC
   42 FORMAT(/,10X,'WRITE ERROR ON LINE',I10)
  145 CONTINUE
C
C        IF LAST READ WAS AN EOF, STOP
   45 IF(IEOF.NE.0) GO TO 50
C
C        IF REACHED MAXIMUM NUMBER OF LINES, STOP
      IF(LINE.GE.NLINES) GO TO 50
C
C        OTHERWISE KEEP GOING
      LINE=LINE+1 $ GO TO 10
C
C
C        FINISH WRITING TAPE
   50 DO 55 I=1,NWORDS
   55 BUF(I)=BACK
   57 IF(NREC.GE.ILIMIT) GO TO 250
      BUFFER OUT(OUT,1) (BUF(1),BUF(NWORDS)) $ NREC=NREC+1
      IF(UNIT(OUT)) 57,58,58
   58 GO TO 57
  250 ENDFILE OUT $ PRINT 51,NPOINTS,LINE
   51 FORMAT(/////,10X,'*** FINISHED OUTPUT OF A',I8,5X,'BY',I8,5X,
     +'RASTER ***')
      PRINT 501
  501 FORMAT(//,15X,'OUTPUT STATISTICS',/)
  502 FORMAT(1X,I7,10X,'INPUT PIXELS WITH INTENSITY',I15,13X,
     .  'FILLED PIXELS =',I9)
      DO 503 I=1,63
      J=I-1
  503 PRINT 502, ICOLOR(I),J,NN(I)
```

```
      PRINT 151
  151 FORMAT(//////1X,110('*')//////)
      PRINT 152, IBLOWUP,IOFX,IOFY
  152 FORMAT(10X,'IMAGE WAS BLOWN UP',I10,10X,'TIMES WITH X OFFSET =',
     .  I10,10X,'AND Y OFFSET =',I10)
      PRINT 153, ILIMIT,NREC $ STOP
  153 FORMAT(///20X,'THIS PRODUCED',  I10,10X,'PIXELS BY',I10,10X,
     .  'LINES FINAL REFERENCE SCENE')
  790 PRINT 91, IBITEX,LENGTH,IWRDEX,IBITPT,IWRDPT
   91 FORMAT(1X,10(1H*),' SUBROUTINE BYTE CALLED WITH INVALID ARGUMENTS
     1',2I5,1X,O20,1X,I5,1X,O20,1X,10(1H*))
      STOP $ END
```

```
      SUBROUTINE SCAN
      COMMON / STUFF / BUF(2040),IRASTER,NRADIUS,ISKIP,IBACK6,NN(64)
      INTEGER BUF
      I=0
    5 I=I+1
      IF(BUF(I).EQ.77B) GO TO 5
      L=I $ I=IRASTER+1
   10 I=I-1
      IF(BUF(I).EQ.77B) GO TO 10
      M=I
      IF((M-L).LE.1) RETURN
      I=L $ J=L $ ISHFLG=0
      DO 100 K=L,M
      IF(BUF(K).LT.77B) GO TO 50
      I=J $ ISHFLG=13 $ GO TO 100
   50 J=K
      IF(ISHFLG.EQ.0) GO TO 100
      ISHFLG=0 $ L1=I+1 $ L2=J-1 $ IAVG=(BUF(I)+BUF(J))/2
      IF((L2-L1+1).GT.ISKIP) GO TO 100
      NN(IAVG+1)=NN(IAVG+1) + (L2-L1+1)
      DO 55 N=L1,L2
   55 BUF(N)=IAVG
  100 CONTINUE
      RETURN $ END
```